(12)特許協力条約に基づいて公開された国際[redacted]

(19) 世界知的所有権機関
国際事務局

(43) 国際公開日
2005年1月6日 (06.01.2005)

PCT

(10) 国際公開番号
WO 2005/001695 A1

(51) 国際特許分類[7]: G06F 12/00, 9/44

(21) 国際出願番号: PCT/JP2004/009043

(22) 国際出願日: 2004年6月21日 (21.06.2004)

(25) 国際出願の言語: 日本語

(26) 国際公開の言語: 日本語

(30) 優先権データ:
特願2003-187690 2003年6月30日 (30.06.2003) JP

(71) 出願人 (米国を除く全ての指定国について): 松下電器産業株式会社 (MATSUSHITA ELECTRIC INDUSTRIAL CO., LTD.) [JP/JP]; 〒5718501 大阪府門真市大字門真１００６番地 Osaka (JP).

(72) 発明者; および
(75) 発明者/出願人 (米国についてのみ): 今西 祐子 (IMANISHI, Yuko). 土井 繁則 (DOI, Shigenori).

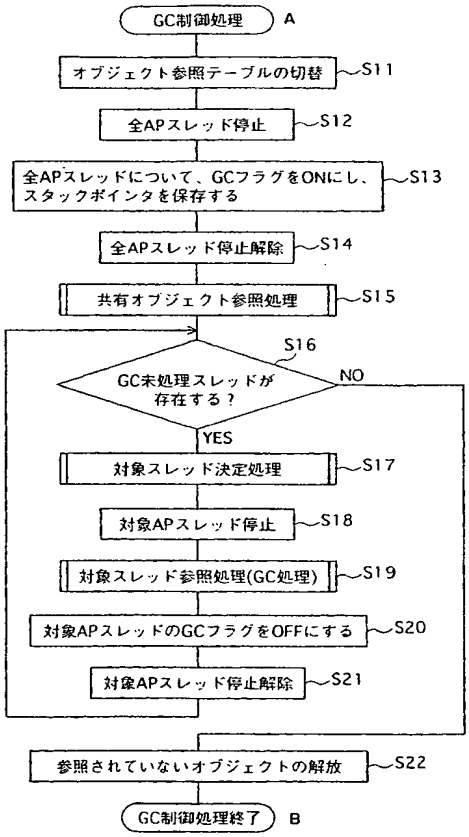
(74) 代理人: 中島 司朗 (NAKAJIMA, Shiro); 〒5310072 大阪府大阪市北区豊崎三丁目２番１号淀川５番館６Ｆ Osaka (JP).

(81) 指定国 (表示のない限り、全ての種類の国内保護が可能): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

[続葉有]

(54) Title: GARBAGE COLLECTION SYSTEM

(54) 発明の名称: ガーベジコレクションシステム



A...GC CONTROL PROCESSING
S11...SWITCH OBJECT REFERENCE TABLE
S12...STOP ALL AP SLEDS
S13...TURN GC FLAG ON FOR ALL AP SLEDS AND SAVE STACK POINTER
S14...RELEASE ALL AP SLED STOP
S15...SHARED OBJECT REFERENCE PROCESSING
S16...GC UNPROCESSED SLED PRESENT?
S17...OBJECT SLED DECISION PROCESSING
S18...STOP OBJECT AP SLED
S19...OBJECT SLED REFERENCE PROCESSING (GC PROCESSING)
S20...TURN OFF GC FLAG OF OBJECT AP SLED
S21...RELEASE OBJECT AP SLED STOP
S22...RELEASE OBJECT NOT REFERENCED
B...GC CONTROL PROCESSING END

(57) Abstract: It is possible to suppress increase of the time required for garbage collection without prolonging the all-sled stop period of AP. A garbage collection system includes: selection means for successively selecting a plurality of sleds in the execution process of the object-orientated program consisting of a plurality of sleds; inspection means for performing inspection processing for stopping the execution of the sled selected, detecting an object accessible from the sled so as to be managed as a non-release object, and resuming execution of the sled; detection means for detecting that an object pointer is made a processing object by the sled in execution after the start of the selection by the selection means and making the object specified by the object pointer a non-release object; and release means for releasing a memory area corresponding to an object other than the object managed as a non-release object after completion of the inspection processing for all the sleds.

[続葉有]

WO 2005/001695 A1

ATTACHMENT A

(84) 指定国(表示のない限り、全ての種類の広域保護が可能): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), ユーラシア (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), ヨーロッパ (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

添付公開書類:
— 国際調査報告書

2文字コード及び他の略語については、定期発行される各PCTガゼットの巻頭に掲載されている「コードと略語のガイダンスノート」を参照。

(57) 要約:

ＡＰの全スレッド停止期間を長期化させず、ガーベジコレクションに要する時間の増加を抑制することを目的とし、複数スレッドで構成されるオブジェクト指向プログラムの実行過程において、複数スレッドを順に選択する選択手段と、選択されたスレッドの実行を停止し、当該スレッドからアクセス可能なオブジェクトを検出して非解放対象として管理し、当該スレッドの実行を再開する検査処理を実施する検査手段と、前記選択手段による前記選択が開始された後において、実行中のスレッドによりオブジェクトポインタが処理対象にされたことを検知した場合には、当該オブジェクトポインタが指すオブジェクトを非解放対処とする検知手段と、前記複数スレッド全てについて前記検査処理が完了した後に非解放対象として管理されているオブジェクト以外のオブジェクトに対応するメモリ領域を開放する解放手段とを備える。

明　細　書

ガーベジコレクションシステム

技術分野

本発明は、アプリケーションプログラム（ＡＰ：Application Program）により使用されたメモリについてのガーベジコレクション（ＧＣ：Garbage Collection）に関する。

背景技術

従来のオブジェクト指向のプログラミング言語には、ＡＰの作成者にメモリ領域の確保や解放を意識させず、ＡＰが使用した後に不要となったオブジェクト（「オブジェクトインスタンス」ともいう。）に対応するメモリ領域の解放を実行環境側に任せる方法を採るものがある。例えばＪａｖａ言語である。なお、Ｊａｖａは米国Ｓｕｎ　Ｍｉｃｒｏｓｙｓｔｅｍｓ社の商標である。

このような言語を用いて記述されたＡＰは、不要となったオブジェクトに対応するメモリ領域を自動的に解放するガーベジコレクション（ＧＣ）機構を備える実行環境上で動かされる。

ＧＣ機構は、ＡＰの実行に際して動的に確保されたオブジェクトに対応するメモリ領域がどこからも参照されていない状態になっていることを検出してそのメモリ領域を解放し再利用可能状態にする。

マルチスレッド構成のＡＰにおいて、各スレッドは、それぞれスタック領域と対応し、動作過程において、スタック領域内にデータを格納したり、格納したデータを参照したり、オブジェクトを生成したりする。オブジェクトを生成した場合には通常はスタック領域内にそのオブジェクトのポインタ、つまりそのオブジェクトのメモリ内における所在位置を指し示すデータ（以下、「オブジェクトポインタ」という。）が格納され、スレッドからのオブジェクトへのアクセスはそのオブジェクトポインタを参照することによりなされる。また、通常、オブジェクトの領域内にも、他のオブジェクトへのオブジェクトポインタが格納される。

ある時点においてＡＰにより参照されている全てのオブジェクトは、ＡＰの各

スレッドに対応する各スタック領域内に含まれるオブジェクトポインタから、直接、又は１以上のオブジェクト内のオブジェクトポインタを媒介にして、辿ることができる。

これに対応してＧＣ機構は、基本的に、ある時点においてスタック領域内のオブジェクトポインタから辿れなくなっているオブジェクトに対応するメモリ領域を不要なものとして解放対象にする。

従来のＧＣ方式として知られているマークアンドスイープ方式は、特定のオブジェクトポインタから辿れる全てのオブジェクトにマークを付けていく処理を行った後に、全てのオブジェクトを走査して、マークの付いていないオブジェクトに対応するメモリ領域を解放する方式である。

このマークアンドスイープ方式のＧＣをマルチスレッド構成のＡＰの実行時に行う場合に、ＧＣの迅速性を最優先して単純に、全スレッドを停止してから、各スレッドに対応するスタック領域内のオブジェクトポインタから辿れる全てのオブジェクトにマークを付与する処理を行い、その後に全スレッドの停止を解除して、マークの付いていないオブジェクトの解放を行うとするならば、次の問題が生じる。

即ち、ＡＰの全スレッドが停止している期間が長期化する可能性があり、この場合に、ユーザの操作等に対して一切反応を示せず、例えばコンピュータのディスプレイの表示内容は変化しないままとなり、ユーザを戸惑わせてしまうという問題である。

この問題を解決する方式として、日本国の特許第３０２７８４５号公報には、このマークアンドスイープを用いたＧＣを、マルチスレッド構成のＡＰを全く停止することなく実行する方式が提案されている。

この方式は、ルートノードなるオブジェクトから辿れる全てのオブジェクト及びスレッド毎の各スタック領域内のオブジェクトポインタから辿れる全てのオブジェクトに対してマークを付与する第１処理を行い、その第１処理中に、ＡＰのスレッド（以下、「ＡＰスレッド」という。）の動作によりオブジェクトへのオブジェクトポインタが移動した場合にそのオブジェクトを表すデータをマークスタックなる領域に積んでおき、そのマークを付与する処理が完了した段階で、更にマークスタックから辿れる全てのオブジェクトに対してマークを付与する第２

処理を行い、最後に、マークされていないオブジェクトに対応するメモリ領域を解放するものである。

しかしながら、上述のＡＰを全く停止せずにマークを付与する方式では、ＡＰスレッドの動作によってスタック領域内のデータが変化するため、第１処理中、スタック領域内のオブジェクトポインタから辿れるオブジェクトに対してマークを付与する処理の一部が無駄となる可能性がある。

例えば、ＧＣを行うスレッドが、ＡＰの１つのスタック領域内のオブジェクトポインタ（ここでは、「オブジェクトポインタＡ」と名付ける。）を検出して、そのオブジェクトポインタＡから辿れるオブジェクトに対してマークを付与する処理（ここでは、「処理Ａ」と名付ける。）を実行している間に、そのスタック領域に対応するＡＰスレッドが、オブジェクトポインタＡ或いはそれらのオブジェクト内の１又は複数のオブジェクトポインタを、コピーしてスタック領域に新たに格納する動作をしたならば、ＧＣを行うスレッドはその処理Ａの終了後にいずれ、再び処理Ａと一部重複する処理を行う、或いはその重複防止のためのチェック処理をわざわざ行うことになり、これが無駄となる。この処理の無駄は、ＧＣの開始から完了までに要するＣＰＵ時間を無駄に増加させることにつながり、ＣＰＵの利用効率を低下させる。

発明の開示

そこで、本発明は、上記問題に鑑みてなされたものであり、ＡＰの全スレッドが停止している期間を長期化させないようにするとともに、ＧＣの開始から完了までに要するＣＰＵ時間の無駄な増加をある程度抑制するようなＧＣ方式を用いるガーベジコレクション（ＧＣ）システムを提供することを目的とする。

上記目的を達成するために、本発明に係るＧＣシステムは、複数スレッドで構成されるオブジェクト指向プログラムの実行過程において不要になったオブジェクトに対応するメモリ領域を解放するＧＣシステムであって、複数スレッドそれぞれを順に選択する選択手段と、選択されたスレッドについて、当該スレッドの実行を停止して、当該スレッドからオブジェクトポインタの参照を通じてアクセス可能であるオブジェクトを検出して当該検出したオブジェクトを非解放対象として管理して、当該スレッドの実行を再開するという手順からなる検査処理を実

施する検査手段と、前記選択手段による前記選択が開始された後において、実行中のスレッドによりオブジェクトポインタが処理対象にされたことを検知した場合には、当該オブジェクトポインタが指すオブジェクトを非解放対象として管理する検知手段と、前記複数スレッド全てについて前記検査処理が完了した後において、非解放対象として管理されているオブジェクト以外のオブジェクトに対応するメモリ領域を、解放する解放手段とを備えることを特徴とする。

ここで、オブジェクトポインタがスレッドにより処理対象にされるということは、ＣＰＵによるスレッドの処理過程においてオブジェクトポインタを処理対象とする命令が実行されるということを意味する。

また、オブジェクトを非解放対象として管理するとは、例えばそのオブジェクトポインタをｆｒｏｍテーブルからｔｏテーブルに移行する後述の方法等により、マーク付与を実現することをいう。

これにより、ＡＰスレッドからスタック内やオブジェクト内のオブジェクトポインタを介して辿ることができるオブジェクトについての非解放対象としての指定の過程つまりマーク付与の過程においては、そのＡＰスレッドを停止させているため、そのＡＰスレッドが動作することに起因してスタック領域内のデータが変化するという事態が起こらないため、その過程におけるマーク付与が無駄になる危険性はなくなり、この点についてはＣＰＵの利用効率低下を防止できる。

また、これにより、全てのＡＰスレッドを停止して上述のマーク付与を行っているのではないため、ＡＰの全スレッドが停止している期間の長期化が防止できる。なお、全ＡＰスレッドを停止させないことによるオブジェクトへの参照状態の変化に対しては、実行中のスレッドによりオブジェクトポインタが処理対象とされたことを監視して対応しているため、いずれかのスレッドからアクセス可能なオブジェクトについては必ず非解放対象として管理されることになる。

また、前記検知手段は、前記検知を、実行中のスレッドについて未だ検査処理がなされていない場合に限り行い、前記検知手段は、前記検知をした場合に、当該実行中のスレッドにより処理対象にされたオブジェクトポインタ、及び当該オブジェクトポインタから辿ることのできるオブジェクト内のオブジェクトポインタを、当該スレッドに対応する作業用メモリ領域に保存する検出部と、前記検査手段によりスレッドの実行が停止されている間に、当該スレッドに対応する作業

用メモリ領域内のオブジェクトポインタから辿ることのできるオブジェクトを非解放対象として管理する管理部とを有することとしてもよい。

これにより、スレッドが後述する参照処理の対象とされるまでの間だけ、つまり、スレッドが、そのスレッドに対応するスタック内等に格納されているオブジェクトポインタが指すオブジェクトへのマーク付与に相当する処理の対象とされるまでの間だけ、インタプリタによるそのスレッドの実行時にオブジェクトポインタがスレッドによる処理対象とされたことを検出して作業用メモリ領域、即ちオブジェクト参照情報のメモリ領域に格納する処理を行うため、一旦スレッドが参照処理の対象とされた後は、そのスレッドはインタプリタによる実行時に、その検出がない分だけ、より高速に動作するようになる。

また、前記検査処理は、選択された当該スレッドに対応するスタック内のオブジェクトポインタが指すオブジェクトを、前記アクセス可能であるとして検出した際において、検出した当該オブジェクトが既に非解放対象として管理されておらず、かつ、当該オブジェクト内にオブジェクトポインタがあるときに限り、当該オブジェクトポインタが指すオブジェクトを更に前記アクセス可能であるとして検出する手順を、繰り返し行うことを内容とする処理であり、前記選択手段は、最初の選択後は、前記検査手段により前記検査処理が行われた後において前記複数スレッドのうち前記検査処理を実施されていないスレッドがある限り更なる選択を行い、前記選択手段は、各スレッドに関する情報を参照し、所定のスレッド選定条件に基づき、前記選択を行うこととしてもよい。

これにより、既に非管理対象として管理されたオブジェクトの配下のオブジェクトを重複的に検出のための処理の対象とすることを防止することができる。また、この重複的に検出のための処理を行わない仕組みにより、参照処理の対象として早期に選択されたスレッドより、遅く選択されたスレッドの方が参照処理に係る時間が短縮される可能性が高まるので、各スレッドに要求される応答性能に鑑みて、予めスレッド選定条件を定めておくことで、各スレッドが適切な応答性能を発揮して実行されるようにある程度制御することができるようになる。

また、前記スレッド選定条件は、スレッド状態がｗａｉｔ状態であるスレッドをスレッド状態がｗａｉｔ状態以外であるスレッドよりも早期に選択することを示す条件を含み、前記選択手段は、前記選択を行う時点でｗａｉｔ状態のスレッ

ドがある限り、当該ｗａｉｔ状態のスレッドを選択することとしてもよい。

これにより、ｗａｉｔ状態のスレッドを停止することになるため、現在動作しているＡＰスレッドに対する悪影響の発生が抑えられる。

また、前記スレッド選定条件は、スレッド優先度のスレッド優先度の低いスレッドをスレッド優先度の高いスレッドよりも早期に選択することを示す条件を含むこととしてもよい。

これにより、スレッド優先度の高いスレッドは、参照処理が短時間でなされる可能性が高くなり、実行性能の低下がある程度防止されるようになる。

また、前記スレッド選定条件は、スレッドに対応するスタックサイズが小さいスレッドをスタックサイズが大きいスレッドよりも早期に選択することを示す条件を含むこととしてもよい。

これにより、スレッドからアクセス可能なオブジェクトへのオブジェクトポインタを格納するスタックの有効範囲が小さい程、参照処理に要する時間は短くなる傾向にあることに鑑みれば、各スレッドの停止時間をある程度均等化する効果が得られ、特定のスレッドの応答性が他と比べて際立って悪くなるという事態がある程度回避できるようになる。

また、前記ガーベジコレクションシステムは、メモリ管理ユニット（ＭＭＵ）を用いてメモリを管理するメモリ管理機構を備え、オブジェクトを生成する必要がある度に、前記メモリ管理機構により当該オブジェクトに対応するメモリ領域を確保し、前記解放手段は、前記メモリ管理機構を介して、メモリ領域の解放を行うこととしてもよい。

これにより、ＭＭＵを用いるメモリ管理機構により確保されているＣ言語その他によるプログラムのデータ領域等と同等の位置付けで、オブジェクトに対するメモリ領域が確保されるため、仮に、特定のヒープ領域を独自に管理してそのヒープ領域内の一部をオブジェクトに対するメモリ領域として確保するような制御を行うこととした場合と比べて、オブジェクトの生成前から無用に大きなヒープ領域を確保しておく必要がない点、そのヒープ領域についてのメモリコンパクション処理をする必要がない点等といった有利な効果が得られる。

図面の簡単な説明

図１は、本発明の実施形態１に係るＧＣシステムの機能ブロック図である。

図２は、オブジェクトとオブジェクトポインタとの関係を例示する図である。

図３は、ｆｒｏｍテーブル及びｔｏテーブルを示す図である。

図４Ａは、参照処理前のｆｒｏｍテーブルの状態を示す図である。

図４Ｂは、参照処理後のｆｒｏｍテーブル及びｔｏテーブルの状態を示す図である。

図５は、スレッド情報、スタック及びオブジェクト参照情報を示す図である。

図６は、参照処理とＡＰスレッドの状態との関係を示す図である。

図７は、スレッド選定条件の内容を示す図である。

図８は、ＧＣ制御処理を示すフローチャートである。

図９は、対象スレッド決定処理を示すフローチャートである。

図１０は、共有オブジェクト参照処理を示すフローチャートである。

図１１は、対象スレッド参照処理を示すフローチャートである。

図１２は、参照処理を示すフローチャートである。

図１３は、命令実行処理を示すフローチャートである。

図１４は、オブジェクトチェーン追跡処理を示すフローチャートである。

図１５は、本発明の実施形態２に係るＧＣシステムの機能ブロック図である。

図１６は、スレッド情報及びスタックを示す図である。

図１７は、実施形態２におけるＧＣ制御処理を示すフローチャートである。

図１８は、実施形態２における命令実行処理を示すフローチャートである。

図１９は、従来のＪａｖａ実行環境により管理されたＪａｖａオブジェクトと、Ｃ言語のプログラムにおけるデータとのメモリ配置を示す図である。

発明を実施するための最良の形態

＜1. 実施形態１＞

以下、本発明の実施形態１に係るガーベジコレクション（ＧＣ）システムについて図面を用いて説明する。

＜1-1. 構成＞

図１は、本発明の実施形態１に係るＧＣシステムの機能ブロック図である。

ＧＣシステム１０は、ＣＰＵ、メモリ等を備えたコンピュータにおいてメモリに格納された制御プログラムがＣＰＵによって実行されることにより実現され、マルチスレッド制御を行う一般のオペレーティングシステム（ＯＳ：Operating System）及びいわゆる仮想マシンを含み、Ｊａｖａ言語等で作成されたアプリケーションプログラムの実行環境として位置づけられるシステムである。

ＧＣシステム１０は、図１に示すように、インタプリタ部１００、オブジェクト管理部２００、スレッド管理部３００及びＧＣ部４００を備える。

ここで、インタプリタ部１００は、基本的にインタプリタであってＡＰを実行する機能を担い、命令実行部１１０及びオブジェクト参照検出部１２０を有する。

オブジェクト管理部２００は、オブジェクトを管理する機能を担い、ｆｒｏｍテーブル２２１、ｔｏテーブル２２２等に対応するメモリ領域であるオブジェクト管理情報記憶部２１０、オブジェクト生成部２３０及びテーブル切替部２４０を有する。

なお、ｆｒｏｍテーブル２２１は、ＧＣ開始前において存在する全オブジェクトに対する全オブジェクトポインタを格納するものと位置づけられるテーブルであり、ｔｏテーブル２２２は、マークアンドスイープ方式でのマーク付与がなされたオブジェクトに対するオブジェクトポインタを格納するために利用されるテーブルである。これらのテーブルについては後に詳細に説明する。

スレッド管理部３００は、マルチスレッド制御を実現し、スレッドを管理する機能を担うものであり、スレッド制御部３１０と、ＡＰスレッド毎に、スレッド情報３２０、オブジェクト参照情報３３０及びスタック３４０に対応するメモリ領域を有する。

なお、スレッド情報３２０は、ＧＣ処理全体の開始時点でＯＮにされそのＡＰスレッドに対しての参照処理が終了するとＯＦＦにされるＧＣフラグを含む。参照処理は、オブジェクトへのマーク付与に相当する処理であり、オブジェクトを参照するためにスタック内等に格納されているオブジェクトポインタと同一内容のオブジェクトポインタをｆｒｏｍテーブルからｔｏテーブルに移動することを主内容とする処理である。

ＧＣ部４００は、基本的にいわゆるガーベジコレクタに相当し、ＧＣ制御部４１０、スレッド選定条件記憶部４２０、参照処理部４３０及び解放部４４０を有

する。このＧＣ部４００を中心として行われるＧＣは、基本的にマークアンドスイープ方式を用い、参照処理の対象となるＡＰスレッドを順番に選択して、選択したＡＰスレッドを停止させて参照処理を行ってからその停止を解除する方式のものである。

インタプリタ部１００における命令実行部１１０は、ＡＰスレッドを構成する命令列を逐次解釈して実行し、命令がオブジェクトの生成命令である場合にはオブジェクト生成部２３０にオブジェクトを生成させる機能を有する。

オブジェクト参照検出部１２０は、ＧＣフラグがＯＮである場合に限って、命令実行部１１０により実行されるスレッド内の命令が、オブジェクトのオブジェクトポインタを処理対象とする命令であるときに、そのオブジェクトポインタを、オブジェクト参照情報３３０のメモリ領域の中に格納する機能を有する。

オブジェクト管理部２００におけるオブジェクト生成部２３０は、いわゆるクラスファイル等のオブジェクトの定義情報を参照して、従来のＯＳのメモリ管理機構に対して、オブジェクトに必要な量のメモリ領域の確保を要求することを通じて、メモリ内にオブジェクトを生成する機能を有する。なお、ＧＣシステム１０は、メモリ管理ユニット（MMU）を利用して物理的なメモリを論理アドレス空間に対応付け、メモリの確保及び解放を管理する従来のＯＳのメモリ管理機構をも含んでおり、このメモリ管理機構は、ある量のメモリ領域の確保が要求されると、論理アドレス空間内の未使用領域のうち、要求された量のメモリ領域を確保したものとして管理し、その確保したメモリ領域の先頭アドレスを返却する機能を有している。従って、オブジェクト生成部２３０は、メモリ管理機構を通じて、オブジェクトに対応する論理アドレス空間内の領域を確保することにより、メモリ内にオブジェクトを生成する。

なお、この論理アドレス空間には、インタプリタ部１００により実行される、Ｊａｖａ言語等で作成されたＡＰにおけるオブジェクト以外に、インタプリタ部１００を介さず従来の一般的なＯＳの直接制御下で実行されるプログラムのデータ、つまり例えばコンパイラ等により既に実行形式にされているプログラムに係るデータ（以下、「Ｎａｔｉｖｅデータ」という。）も配置される。このインタプリタ部１００を介さずに実行可能な形式のプログラムが、Ｎａｔｉｖｅデータを配置するために従来のＯＳのメモリ管理機構を通じてメモリ領域を確保し、そ

こにＮａｔｉｖｅデータを配置するのと同様の仕組みで、オブジェクト生成部２３０により生成されるオブジェクトについてのメモリ領域の確保はなされる。

かかるＮａｔｉｖｅデータやそれに係るプログラムは、ＧＣシステム１０の特徴となるＧＣの動作とは直接関係しないので、ここでは詳しい説明を省略する。

なお、上述したＡＰスレッドとは、インタプリタ部１００を介して実行されるＡＰの実行単位となるスレッドをいい、つまりＮａｔｉｖｅデータに係るプログラムの実行単位となるスレッド以外のスレッドをいう。また、ここでいうオブジェクトにはＮａｔｉｖｅデータは含まれない。

テーブル切替部２４０は、ｆｒｏｍテーブルを指すものとして用いられているポインタとｔｏテーブルを指すものとして用いられているポインタとを交換することにより、瞬時にｆｒｏｍテーブルの内容とｔｏテーブルの内容とを交換したに等しい効果を生じさせる機能を有する。

スレッド管理部３００におけるスレッド制御部３１０は、マルチスレッド制御を行い、ＡＰを構成する各ＡＰスレッド及びＧＣ処理のスレッドを含む各スレッドを並列的に実行する。即ち、スレッド制御部３１０は、微小時間毎に各スレッドを切り替えて実行する。なお、スレッド管理部３００は、Ｎａｔｉｖｅデータに係るプログラムのスレッドも並列的に実行するが、このマルチスレッド制御の機能は、基本的に従来のＯＳ等が有する機能であり、ＧＣシステム１０の特徴となるＧＣの動作とは直接関係しないので、ここでは、主に、ＡＰスレッド及びＧＣ処理のスレッドにのみ着目して説明する。

ＧＣ部４００におけるスレッド選定条件記憶部４２０は、参照処理を行う対象となるＡＰスレッドを選択するための条件を示すスレッド選定条件を記憶しているメモリ領域である。

参照処理部４３０は、参照処理を行う機能を有し、解放部４４０は、マーク付与がなされていないオブジェクトの解放、即ち、ＧＣの終了段階においてｆｒｏｍテーブル２２１に残存しているオブジェクトポインタが指すオブジェクトに対応するメモリ領域を解放する機能を有する。このメモリ領域の解放は、オブジェクトポインタつまりオブジェクトの論理アドレスを指定して、ＭＭＵを利用した従来のＯＳのメモリ管理機構に対する解放要求を行うことにより実現される。なお、ＧＣシステム１０に含まれる従来のＯＳのメモリ管理機構は、論理アドレス

を指定した解放要求に対して、その論理アドレスと対応付けて管理している確保領域を、新たにメモリを確保する必要がある際に割り当てることができる未使用領域として管理する。

また、ＧＣ制御部４１０は、ＧＣ制御処理を実行する機能を有する。即ち、ＧＣ制御部４１０は、スレッド選定条件を参照することにより、参照処理の対象にするＡＰスレッドを１つ選定し、選択したＡＰスレッドについて、スレッド制御部３１０にそのＡＰスレッドを停止させてから、参照処理部４３０にそのＡＰスレッドに対応するスタック及びオブジェクト参照情報に基づいて参照処理を行わせ、その後にスレッド制御部３１０に選択したＡＰスレッドの停止を解除させてから、次のＡＰスレッドの選択を行うという手順を、未処理のＡＰスレッドが存在しなくなるまで繰り返してから、解放部４４０に、マーク付与がなされていないオブジェクトの解放を行わせる機能を有する。

＜1-2. データ＞

以下、ＧＣシステム１０において取り扱われるデータについて説明する。

図２は、オブジェクトとオブジェクトポインタとの関係を例示する図である。

同図では、オブジェクトポインタを、小さい塗り潰し円で表現している。

オブジェクトを指すつまりメモリ５００に対応する論理アドレス空間に配置されたオブジェクトの所在位置を指すオブジェクトポインタは、オブジェクト内、スタック内、又は共有オブジェクト管理情報３５３内に存在し得る。ここで、共有オブジェクト管理情報３５３は、ＡＰの実行環境としての仮想マシンにおいて必要なため確保されているオブジェクト群に対するオブジェクトポインタ群を含むデータである。

図２の例では、共有オブジェクト管理情報３５３内のあるオブジェクトポインタがオブジェクト２０１ｃを指している。

また、ＡＰスレッド３５１に対応するスタック内のあるオブジェクトポインタは、オブジェクト２０１ａを指しており、つまり、オブジェクト２０１ａのメモリアドレスを内容としており、また別のオブジェクトポインタは、オブジェクト２０１ｄを指している。従って、ＡＰスレッド３５１は、実行中にこれらのオブジェクトポインタを参照することによりオブジェクト２０１ａ、オブジェクト２０１ｄにアクセス可能な状態である。

また、ＡＰスレッド３５２に対応するスタック内のあるオブジェクトポインタは、オブジェクト２０１ｂを指しており、オブジェクト２０１ｂ内のデータとして、オブジェクト２０１ｅを指すオブジェクトポインタが含まれている。従って、ＡＰスレッド３５２は、これらのオブジェクトポインタを辿ることによりオブジェクト２０１ｅにアクセス可能な状態である。

また、図２は、インタプリタ部１００を介して実行されるＪａｖａ言語等で作成されたＡＰに対応するオブジェクト２０１ａ～２０１ｅ以外にも、同じくＯＳのメモリ管理機構により領域が確保されたものであるＮａｔｉｖｅデータ５０１ａ～５０１ｄがメモリ５００内に混在していることをも表している。即ち、オブジェクト２０１ａ～２０１ｅは、まとめて、ある種のヒープ領域中に置かれているのではなく、各Ｎａｔｉｖｅデータと同様に、ＯＳのメモリ管理機構によりそれぞれ個別に領域が割り当てられている。

なお、図２には示さなかったが、スレッド情報３２０、スタック３４０及びオブジェクト参照情報３３０もメモリ５００内に確保されるものであり、また、オブジェクト管理情報記憶部２１０は、メモリ５００中の一部に相当する。なお、インタプリタ部１００を介して実行されるＡＰのプログラム部分や、Ｎａｔｉｖｅデータにアクセスするプログラム部分は、読み書き可能なメモリ（ＲＡＭ）中に置かれるものとしても、読み出しだけ可能なメモリ（ＲＯＭ）に置かれるものとしてもよい。

図３は、ｆｒｏｍテーブル及びｔｏテーブルを示す図である。

メモリ５００内のオブジェクト管理情報記憶部２１０内に、オブジェクトポインタ又はｎｕｌｌ値を十分な数だけ格納するための２つのテーブルが設けられており、また、ｆｒｏｍテーブルポインタ２１１及びｔｏテーブルポインタ２１２が存在する。ｆｒｏｍテーブルポインタ２１１は、その２つのテーブルのうち一方を指すポインタであり、ｔｏテーブルポインタ２１２は、他方を指すポインタである。ここでは、現時点でｆｒｏｍテーブルポインタ２１１が指しているテーブルをｆｒｏｍテーブルと称しており、ｔｏテーブルポインタ２１２が指しているテーブルをｔｏテーブルと称している。

ｆｒｏｍテーブル２２１は、ＧＣ開始時点においては、その時に存在する全オブジェクトに対する全オブジェクトポインタを格納している。

また、ｔｏテーブル２２２は、参照処理の過程でｆｒｏｍテーブル２２１内から移動してオブジェクトポインタを格納するテーブルである。なお、ｆｒｏｍテーブル２２１中のオブジェクトポインタがｔｏテーブル２２２へと移動されるときには、そのオブジェクトポインタが格納されていたｆｒｏｍテーブル２２１内の位置にあたるメモリ内容はｎｕｌｌ値にされる。

図４Ａ及び図４Ｂは、参照処理によるｆｒｏｍテーブル及びｔｏテーブルの内容の変化を示すための図であり、図４Ａは参照処理開始前の状態を示し、図４Ｂは参照処理後の状態を示す。

図４Ａでは、参照処理開始前におけるｆｒｏｍテーブル２２１ｘは、ｏｂｊＡ２０２ａ、ｏｂｊＢ２０２ｂ、ｏｂｊＣ２０２ｃのそれぞれのオブジェクトを指すオブジェクトポインタを含んでいる。なお、図４Ａ及び図４Ｂでは、各オブジェクトとＮａｔｉｖｅデータとがメモリ５００内に混在していることをも表現している。

その後にＡＰスレッド３５４を対象として参照処理が行われた場合には、ＡＰスレッド３５４のスタック内に含まれているオブジェクトポインタと同値のものがｆｒｏｍテーブルからｔｏテーブルに移動され、その結果として図４Ｂに示す状態になる。ｆｒｏｍテーブル２２１ｙでは、ｏｂｊＡ２０２ａを指していたオブジェクトポインタの存在した場所と、ｏｂｊＣ２０２ｃを指していたオブジェクトポインタの存在した場所との両方がｎｕｌｌ値に更新されており、ｔｏテーブル２２２ｙには、ｏｂｊＡ２０２ａを指すオブジェクトポインタと、ｏｂｊＣ２０２ｃを指すオブジェクトポインタとが格納されている。

図５は、スレッド情報、スタック及びオブジェクト参照情報を示す図である。

スレッド情報３２０は、ＡＰスレッド毎に、そのスレッド生成時に生成され、そのＡＰスレッドについての情報を含むものであり、具体的には、状態３２１、優先度３２２、スタックポインタ３２３、ＧＣ開始時スタックポインタ３２４、ＧＣフラグ３２５、オブジェクト参照情報先頭ポインタ３２６及びオブジェクト参照情報カレントポインタ３２７を含む。なお、ＡＰスレッド生成時には、スレッド情報３２０のメモリ領域のみならずスタック３４０のメモリ領域も確保される。

スレッド情報３２０における状態３２１は、マルチスレッド制御のためのスレ

ッド状態を示す情報であり、ｗａｉｔ状態、ｒｕｎ状態、ｒｅａｄｙ状態等の別を示す。

優先度３２２は、スレッドの優先度を示す情報であり、優先度は例えばスレッド生成時にＡＰからの指定を受けて定まる。なお、マルチスレッド制御において優先度が高いスレッドがｒｅａｄｙ状態であればそれより優先度の低いスレッドよりも優先的にｒｕｎ状態にされる。

スタックポインタ３２３は、該当のスレッドについてのスタック内における現在の有効なデータ範囲の終端を示すものである。なお、マルチスレッド制御において、スレッドがｒｕｎ状態から、他の状態つまり停止した状態に切り替えられる際に、それまでスタックポインタを指す所定レジスタに格納されていた値がこのスタックポインタ３２３に格納され、スレッドがｒｕｎ状態に切り替えられる際に、このスタックポインタ３２３内の値がその所定レジスタに設定される。

ＧＣ開始時スタックポインタ３２４は、ＧＣ開始時におけるスタック内における有効なデータ範囲の終端を示すものである。

ＧＣフラグ３２５は、ＧＣ処理全体の開始時点でＯＮにされ、そのＡＰスレッドに対しての参照処理が終了するとＯＦＦにされる。

オブジェクト参照情報先頭ポインタ３２６は、該当ＡＰスレッドに対応するオブジェクト参照情報のメモリ領域の先頭を示すものであり、ＡＰスレッド生成の際のそのメモリ領域確保時に設定される。

また、オブジェクト参照情報カレントポインタ３２７は、該当ＡＰスレッドに対応するオブジェクト参照情報のメモリ領域において、オブジェクト参照検出部１２０が次にオブジェクトポインタを格納すべき位置を示す情報であり、オブジェクト参照検出部１２０よって参照及び更新される。

図６は、参照処理とＡＰスレッドの状態との関係を示す図である。

ＧＣスレッド３５６は、ＧＣ制御部４１０によるＧＣを実行するスレッドであり、ＧＣとして各ＡＰスレッドを順番に対象として参照処理を行う。ここでは、ＡＰスレッド３５５ａ、ＡＰスレッド３５５ｂ、ＡＰスレッド３５５ｃの順に参照処理が行われる場合の例を示している。

ＡＰスレッド３５５ａは、参照処理がなされた後の状態であり、実行中である。ここでいう実行中は、ｓｌｅｅｐ状態つまり停止状態ではないことであり、各瞬

時においてはｒｕｎ状態やｒｅａｄｙ状態等と変化し得る。

ＡＰスレッド３５５ｂは、参照処理がなされているところの状態であり、停止している。参照処理は、ＡＰスレッドを強制的に停止状態にして、スタック及びオブジェクト参照情報を参照して行われる。

ＡＰスレッド３５５ｃは、参照処理が未だなされていない状態であり、実行中である。

図７は、スレッド選定条件の内容を示す図である。

スレッド選定条件４２１は、どのＡＰスレッドを優先して参照処理の対象とするかの判断基準となる情報であり、スレッド状態４２２、スレッド優先度４２３及びスタックサイズ４２４から構成される。

スレッド状態４２２は、どの状態のＡＰスレッドを優先するかを示す情報であり、図７の例はｗａｉｔ状態のＡＰスレッドを、ｒｕｎ状態等のＡＰスレッドより優先する旨を示している。

スレッド優先度４２３は、ＡＰスレッドについて定められている優先度の高いものを参照処理の対象として優先的に選定するか、又は低いものを優先的に選定するかを示す情報であり、図７の例では優先度の低いものを優先的に選定する旨を示している。

また、スタックサイズ４２４は、ＡＰスレッドに対応するスタック領域の有効範囲の大きさが、大きいところのＡＰスレッドを優先的に選定するか、又は小さいところのＡＰスレッドを優先的に選定するかを示す情報であり、図７の例ではスタック領域の有効範囲の大きさが小さいものを優先的に選定する旨を示している。

＜1-3. 動作＞

以下、ＧＣシステム１０の動作について説明する。

図８は、ＧＣ制御処理を示すフローチャートである。

ＧＣ制御処理は、タイマーに基づき一定周期で行われる。

ＧＣ制御部４１０は、最初に、ｆｒｏｍテーブルポインタ２１１とｔｏテーブルポインタ２１２の内容を交換することにより、ｆｒｏｍテーブルとｔｏテーブルとの内容を切り替える（ステップＳ１１）。これにより、現在解放されていない全てのオブジェクトを指す全てのオブジェクトポインタがｆｒｏｍテーブルに

格納されている状態になる。

続いて、ＧＣ制御部４１０は、スレッド制御部３１０に全ＡＰスレッドを停止させ（ステップＳ１２）、全ＡＰスレッドについてのスレッド情報中のＧＣフラグ３２５をＯＮにし、現在のスタックポインタをスレッド情報中のＧＣ開始時スタックポインタ３２４に設定し（ステップＳ１３）、スレッド制御部３１０に全ＡＰスレッドの停止を解除させる（ステップＳ１４）。ステップＳ１３においては、更に、ＡＰスレッド毎に、オブジェクト参照情報のメモリ領域を確保し、そのメモリ領域の先頭を指すポインタをオブジェクト参照情報先頭ポインタ３２６及びオブジェクト参照情報カレントポインタ３２７に設定する。

なお、スレッド制御部３１０は、従来のＯＳのマルチスレッド制御機構と同様の方式によりスレッドの停止及び停止解除を行う。このスレッドの停止は、スレッドを停止状態、即ちｓｌｅｅｐ状態にする制御であり、スレッドの停止解除とは、ｓｌｅｅｐ状態を解除して、ｒｅａｄｙ状態に戻す制御である。

ステップＳ１４に続いて、ＧＣ制御部４１０は、共有オブジェクト管理情報３５３内のオブジェクトポインタを辿ることで到達するオブジェクトに対するマーク付与を行うための共有オブジェクト参照処理を行い（ステップＳ１５）、ＧＣ未処理スレッドつまり未だ参照処理の対象として選定されていないＡＰスレッドが存在するか否かを判定する（ステップＳ１６）。なお、共有オブジェクト参照処理については後述する。

ステップＳ１６においてＧＣ未処理スレッドが存在すると判定した場合には、ＧＣ制御部４１０は、参照処理の対象となるＡＰスレッドを決定するための対象スレッド決定処理を行い（ステップＳ１７）、スレッド制御部３１０にその決定したＡＰスレッドを停止させ（ステップＳ１８）、その決定したＡＰスレッドに関しての参照処理を内容とする対象スレッド参照処理を実行し(ステップＳ１９)、その決定したＡＰスレッドのＧＣフラグ３２５をＯＦＦにし(ステップＳ２０)、スレッド制御部３１０にその決定したＡＰスレッドの停止を解除させて（ステップＳ２１）、再度、ステップＳ１６の判定に戻る。なお、対象スレッド決定処理及び対象スレッド参照処理については、後述する。また、ステップＳ２０の直後に、ＧＣ制御部４１０は、対象のＡＰスレッドに対応するオブジェクト参照情報のメモリ領域を解放する。

また、ステップＳ１６において、ＧＣ未処理スレッドが存在しないと判定した場合には、ＧＣ制御部４１０は、解放部４４０にマーク付与がなされていないオブジェクトを解放させ（ステップＳ２２）、ＧＣ制御処理を終える。ステップＳ２２においては、解放部４４０は、参照処理によってｔｏテーブルにオブジェクトポインタが移行されていないところのオブジェクト、つまりｆｒｏｍテーブルに残存しているオブジェクトポインタが指すオブジェクトに対応するメモリ領域を解放する。この解放は、オブジェクト生成部２３０が、オブジェクト生成のためにオブジェクトに割り当てるためのメモリ領域を、ＭＭＵを利用したＯＳのメモリ管理機構に基づき確保することに対応し、その確保されたメモリ領域の解放を意味する。なお、この解放されたメモリ領域は、メモリ管理機構により、新たにオブジェクトやＮａｔｉｖｅデータの格納域として割り当てられ得るようになる。

図９は、対象スレッド決定処理を示すフローチャートである。

ＧＣ制御部４１０は、スレッド選定条件記憶部４２０内のスレッド選定条件４２１を参照して、対象スレッド決定処理を行う。

まず、ＧＣ制御部４１０は、各ＡＰスレッドに対応するスレッド情報を参照して、状態３２１がｗａｉｔ状態のＡＰスレッドを検索する（ステップＳ３１）。この検索結果のスレッド数を判定し（ステップＳ３２）、１であれば、その検索結果のＡＰスレッドを参照処理の対象スレッドとして決定し(ステップＳ３７)、対象スレッド決定処理を終える。

また、ステップＳ３２において検索結果のスレッド数が０であれば、ＧＣ制御部４１０は、スレッドの優先度３２２が最も低いＡＰスレッドを検索し（ステップＳ３３）、検索結果のスレッド数は１か否かを判定する（ステップＳ３５）。また、ステップＳ３２において検索結果のスレッド数が２以上であれば、検索結果のＡＰスレッドの中から更に絞り込むため、スレッドの優先度３２２が最も低いＡＰスレッドを検索し（ステップＳ３４）、検索結果のスレッド数は１か否かを判定する（ステップＳ３５）。

ステップＳ３５においてスレッド数が１と判定した場合には、ＧＣ制御部４１０は、その検索結果のＡＰスレッドを参照処理の対象スレッドとして決定し（ステップＳ３７）、対象スレッド決定処理を終える。

また、ステップＳ３５において、検索結果のスレッド数が１でないと判定した場合、即ち、最も低い優先度のＡＰスレッドが複数存在した場合には、ＧＣ制御部４１０は、検索結果のＡＰスレッドの中からスタックサイズが最小のＡＰスレッドを検索し（ステップＳ３６）、スタックサイズが最小のＡＰスレッドを参照処理の対象スレッドとして決定し（ステップＳ３７）、対象スレッド決定処理を終える。

図１０は、共有オブジェクト参照処理を示すフローチャートである。

ＧＣ制御部４１０は、オブジェクトポインタ群を含む共有オブジェクト管理情報３５３の先頭のオブジェクトポインタに着目して（ステップＳ４１）、参照処理部４３０に参照処理（ステップＳ４２）を行わせることにより、ＡＰの実行環境としての仮想マシンにおいて必要なため確保されているオブジェクト群全てにマーク付与を行う。

図１１は、対象スレッド参照処理を示すフローチャートである。

ＧＣ制御部４１０は、対象スレッドに対応するスレッド情報中のＧＣ開始時スタックポインタ３２４が指す位置に着目し（ステップＳ５１）、参照処理部４３０に参照処理（ステップＳ５２）を行わせることにより、スタック領域内のオブジェクトポインタから辿ることのできる全てのオブジェクトへのマーク付与を行う。なお、ＧＣ制御処理（図８参照）の開始時点と、この対象スレッド参照処理を開始する時点は異なるため、ＧＣ開始時スタックポインタ３２４が指す位置とそれに続く各位置の内容は、ＧＣ開始時点とは異なっているが、後続するステップＳ５３及びステップＳ５４によって、ＧＣ開始以後に、対象スレッドの動作によって変化したオブジェクトポインタから辿ることのできる全てのオブジェクトへのマーク付与が行われるので、過剰にマーク付与をする場合はあり得るが、参照されているにも関わらずマーク付与が漏れるという事態は生じない。

続いて、ＧＣ制御部４１０は、オブジェクト参照情報先頭ポインタ３２６の指す位置に着目し（ステップＳ５３）、参照処理部４３０に参照処理（ステップＳ５４）を行わせ、対象スレッド参照処理を終える。

図１２は、参照処理を示すフローチャートである。

参照処理部４３０は、着目されているメモリ位置から、オブジェクトポインタを検索し（ステップＳ６１）、オブジェクトポインタを検出したか否かを判定し

（ステップＳ６２）、検出した場合にはその検出したオブジェクトポインタと同値のオブジェクトポインタがｆｒｏｍテーブルに存在するか否かを判定し（ステップＳ６４）、ｆｒｏｍテーブルに存在すれば、ｆｒｏｍテーブルからｔｏテーブルにそのオブジェクトポインタをコピーして、ｆｒｏｍテーブル内のそのオブジェクトポインタが存在していた位置にｎｕｌｌ値を記録し(ステップＳ６６)、そのオブジェクトポインタが指すオブジェクトのデータ先頭に着目し（ステップＳ６７）、再度ステップＳ６１に戻る。

また、ステップＳ６４において、ｆｒｏｍテーブルにはそのオブジェクトポインタが存在しないと判定した場合には、既にｆｒｏｍテーブルからｔｏテーブルに移されているため、参照処理部４３０は、着目している位置の次の位置に着目し（ステップＳ６５）、再度ステップＳ６１に戻る。ステップＳ６５及びステップＳ６１の組み合わせにより、共有オブジェクト管理情報３５３内のオブジェクトポインタを次々と検索することや、スタック内のオブジェクトポインタを次々と検索することや、オブジェクト参照情報内のオブジェクトポインタを次々と検索することや、あるオブジェクト内のデータメンバとしてのオブジェクトポインタを次々と検索すること等が実現される。

また、ステップＳ６２においてオブジェクトポインタを検出できなかったと判定した場合には、参照処理部４３０は、着目位置がオブジェクト内であるか否かを判定する（ステップＳ６３）。なお、オブジェクトポインタを検出できなかったというのは、着目位置が共有オブジェクト管理情報３５３内であればその共有オブジェクト管理情報３５３内においてオブジェクトポインタを検出できなかったということであり、着目位置がスタック内であればスタック内においてオブジェクトポインタを検出できなかったということであり、着目位置がオブジェクト参照情報内であればそのオブジェクト参照情報内においてオブジェクトポインタを検出できなかったということであり、着目位置がオブジェクト内であればそのオブジェクト内においてオブジェクトポインタを検出できなかったということである。

ステップＳ６３において着目位置がオブジェクト内であると判定した場合には、参照処理部４３０は、そのオブジェクト内に着目前の着目位置の次位置に着目し（ステップＳ６８）、再度ステップＳ６１に戻る。このステップＳ６８によって、

参照処理部４３０は、ステップＳ６７によってオブジェクト内に着目する前に着目していたオブジェクトポインタの次の位置に着目することになる。

また、ステップＳ６３において着目位置がオブジェクト内ではないと判定した場合には、参照処理部４３０は、参照処理を終える。

図１３は、命令実行処理を示すフローチャートである。

インタプリタ部１００における命令実行部１１０は、ｒｕｎ状態にされたＡＰスレッドにおけるプログラム中の命令記述を逐次解釈して実行する命令実行処理を行う。

まず、命令実行部１１０は、ＡＰスレッドの現在の実行位置における命令を解釈して実行し（ステップＳ７１）、オブジェクト参照検出部１２０は、そのＡＰスレッドに対応するスレッド情報中のＧＣフラグ３２５がＯＮであるか否かを判定し（ステップＳ７２）、ＯＮでなければ、命令実行部１１０が次の命令の解釈実行を行う（ステップＳ７１）。

ステップＳ７２において、ＧＣフラグ３２５がＯＮであると判定した場合には、オブジェクト参照検出部１２０は、ステップＳ７１において命令実行部１１０により実行された命令が、オブジェクトポインタを処理対象とする命令であったか否かを判定し（ステップＳ７３）、オブジェクトポインタを処理対象とする命令でなかったならば、命令実行部１１０が次の命令の解釈実行を行う（ステップＳ７１）。なお、オブジェクトポインタを処理対象とする命令とは、スタック内のオブジェクトポインタをオブジェクト内にコピーする演算を指示内容とする命令や、オブジェクト内のオブジェクトポインタを他のオブジェクト内にコピーする演算を指示内容とする命令である。

ステップＳ７３において、オブジェクトポインタを処理対象とする命令であったと判定した場合には、オブジェクト参照検出部１２０はそのオブジェクトポインタと同値のものがオブジェクト参照情報領域に既に格納済みであるか否かを判定し（ステップＳ７４）、格納済みである場合には、命令実行部１１０が次の命令の解釈実行を行う（ステップＳ７１）。

ステップＳ７４において、そのオブジェクトポインタと同値のものがオブジェクト参照情報のメモリ領域に格納済みでない場合には、オブジェクト参照検出部１２０は、そのオブジェクトポインタを、オブジェクト参照情報カレントポイン

タ３２７が指すオブジェクト参照情報中の位置に格納し（ステップＳ７５）、そのオブジェクトポインタの指すオブジェクトに着目して、オブジェクトチェーン追跡処理（ステップＳ７６）を行い、その後は命令実行部１１０が次の命令の解釈実行を行う（ステップＳ７１）。

なお、ステップＳ７１における実行対象とされる命令がオブジェクトを生成する命令である場合には、命令実行部１１０は、オブジェクト生成部２３０にオブジェクトの生成を指示し、これを受けてオブジェクト生成部２３０は、オブジェクトを生成するとともにそのオブジェクトを指すオブジェクトポインタをＡＰスレッドに対して返却し、そのオブジェクトポインタのコピーをｔｏテーブル内に設定する。

図１４は、オブジェクトチェーン追跡処理を示すフローチャートである。

オブジェクト参照検出部１２０は、着目しているオブジェクトに未着目のオブジェクトポインタが存在するか否かを判定し（ステップＳ８１）、存在する場合には、その未着目の１つのオブジェクトポインタに着目し（ステップＳ８２）、その着目したオブジェクトポインタをオブジェクト参照情報のメモリ領域に格納し（ステップＳ８３）、その着目したオブジェクトポインタの指すオブジェクトに着目して更にオブジェクトチェーン処理（ステップＳ８１～Ｓ８４）を行う（ステップＳ８４）。なお、オブジェクトポインタをオブジェクト参照情報のメモリ領域に格納したときには、オブジェクト参照検出部１２０は、オブジェクト参照情報カレントポインタをそのオブジェクトポインタのサイズ分だけ進める。

また、ステップＳ８１において、着目しているオブジェクトに未着目のオブジェクトポインタが存在していないと判定した場合には、オブジェクト参照検出部１２０は、オブジェクトチェーン追跡処理を終える。

従って、この命令実行処理及びオブジェクトチェーン追跡処理によって、ＧＣフラグ３２５がＯＮである場合において、オブジェクトポインタが処理対象とされたときにはそのオブジェクトポインタから辿ることのできるオブジェクトポインタはいずれもオブジェクト参照情報のメモリ領域に格納されることになる。

なお、ＡＰスレッドによるオブジェクトポインタを処理対象とする演算の実行によって、そのオブジェクトポインタで指されていたオブジェクトにそのＡＰスレッドからアクセスすることが不可能な状態となった場合、つまり、ＡＰスレッ

ドの動作により、（ａ）オブジェクトポインタがクリアされた場合、（ｂ）オブジェクトポインタが他の内容に更新された場合、或いは、（ｃ）オブジェクトポインタがスタック領域の有効範囲内に存在する場合においてスタックポインタの変更がなされてスタック領域の有効範囲外となりＡＰスレッドから基本的にアクセス不能となった場合にも、そのＡＰスレッドによって予めそのオブジェクトポインタが他のＡＰスレッドからはアクセス可能な場所にコピーされていたならば、そのオブジェクトポインタで指されるオブジェクトにはマーク付与が必要となるので、そのために、上述したステップＳ７２～Ｓ７６、ステップＳ８１～Ｓ８４の処理が行われる。

＜2.実施形態２＞

以下、本発明の実施形態２に係るＧＣシステムについて図面を用いて説明する。

実施形態２に係るＧＣシステムは、ＡＰスレッドを順次停止して参照処理を行う点で、実施形態１に係るＧＣシステム１０と基本的に同様である。ただし、実施形態２に係るＧＣシステムは、主に、ＡＰスレッド毎にオブジェクト参照情報やＧＣフラグを有するのではなくシステム全体でまとめてオブジェクト参照情報やＧＣフラグを有する点や、スレッド情報の内容を一般的なマルチスレッド制御に要する程度に縮小している点が、ＧＣシステム１０と異なる。

図１５は、本発明の実施形態２に係るＧＣシステムの機能ブロック図である。

図１５に示すＧＣシステム２０の構成要素のうち、実施形態１で示したＧＣシステム１０と同様の構成要素については、図１と同じ符号を付しており、ここではＧＣシステム２０に特有の点を中心に説明する。なお、特に説明しない点については、ＧＣシステム２０はＧＣシステム１０と同様である。

ＧＣシステム２０は、図１５に示すように、インタプリタ部１１００、オブジェクト管理部２００、スレッド管理部１３００及びＧＣ部１４００を備える。

ここで、インタプリタ部１１００は、基本的にインタプリタであってＡＰを実行する機能を担い、命令実行部１１０及びオブジェクト参照検出部１１２０を有する。

スレッド管理部１３００は、マルチスレッド制御を行いスレッドを管理する機能を担い、スレッド制御部３１０と、ＡＰスレッド毎に、スレッド情報１３２０及びスタック３４０に対応するメモリ領域を有する。なお、スレッド情報１３２

０は、図１６に示すように状態３２１、優先度３２２及びスタックポインタ３２３を含むものである。

ＧＣ部１４００は、基本的にいわゆるガーベジコレクタに相当し、ＧＣ制御部１４１０と、スレッド選定条件記憶部４２０と、参照処理部１４３０と、解放部４４０と、オブジェクト参照情報１４５０及びＧＣフラグ１４６０に対応するメモリ領域とを有する。

インタプリタ部１１００におけるオブジェクト参照検出部１２０は、ＧＣフラグ１４６０がＯＮである場合に限って、命令実行部１１０により実行されるＡＰスレッド内の命令が、オブジェクトのオブジェクトポインタを処理対象とする命令であるときに、そのオブジェクトポインタを、オブジェクト参照情報１４５０のメモリ領域の中に格納する機能を有する。

ＧＣ部１４００を中心として行われるＧＣは、基本的にマークアンドスイープ方式を用い、参照処理の対象となるＡＰスレッドを順番に選択して、選択したＡＰスレッドを停止させて参照処理を行ってからその停止を解除する方式のものである。

ＧＣ部１４００における参照処理部１４３０は、スタック及びオブジェクト参照情報１４５０を参照することにより参照処理を行う機能を有する。オブジェクト参照情報１４５０は、ＧＣシステム１０におけるオブジェクト参照情報と同様にオブジェクトポインタを内容とする情報であるが、ＡＰスレッド毎に存在するのではない。このオブジェクト参照情報１４５０のメモリ領域には、全ＡＰスレッドについてのオブジェクト参照検出部１１２０により、オブジェクトポインタが命令の処理対象とされた場合にそのオブジェクトポインタが格納される。なお、スレッド情報１３２０、スタック３４０、オブジェクト参照情報１４５０、ＧＣフラグ１４６０は、オブジェクトやＮａｔｉｖｅデータと同様に、オブジェクト管理情報記憶部２１０を含むメモリに格納される。

また、ＧＣ制御部１４１０は、図１７に示すＧＣ制御処理を実行する機能を有する。

図１７は、実施形態２におけるＧＣ制御処理を示すフローチャートである。なお、このフローチャート中の処理ステップのうち、実施形態１のＧＣ制御処理と同様のものについては、図８と同じ符号を付して示している。

このＧＣ制御処理は、タイマーに基づき一定周期で行われる。

ＧＣ制御部１４１０は、最初に、ｆｒｏｍテーブルポインタ２１１とｔｏテーブルポインタ２１２の内容を交換することにより、ｆｒｏｍテーブルとｔｏテーブルとの内容を切り替え（ステップＳ１１）、ＧＣフラグ１４６０をＯＮにする（ステップＳ１１１）。

ステップＳ１１１に続いて、ＧＣ制御部１４１０は、共有オブジェクト管理情報３５３内のオブジェクトポインタを辿ることで到達するオブジェクトに対するマーク付与を行うための共有オブジェクト参照処理を行い（ステップＳ１５）、ＧＣ未処理スレッドつまり未だ参照処理の対象として選定されていないＡＰスレッドが存在するか否かを判定する（ステップＳ１６）。

ステップＳ１６においてＧＣ未処理スレッドが存在すると判定した場合には、ＧＣ制御部１４１０は、参照処理の対象となるＡＰスレッドを決定するための対象スレッド決定処理を行い（ステップＳ１７）、スレッド制御部３１０にその決定したＡＰスレッドを停止させ（ステップＳ１８）、その決定したＡＰスレッドに対応するスタックポインタの位置に着目して（ステップＳ１１２）、参照処理部１４３０に参照処理（図１２参照）を実行させ（ステップＳ１１３）、スレッド制御部３１０にその決定したＡＰスレッドの停止を解除させて（ステップＳ２１）、再度、ステップＳ１６の判定に戻る。

また、ステップＳ１６において、ＧＣ未処理スレッドが存在しないと判定した場合には、ＧＣ制御部１４１０は、スレッド制御部３１０に全ＡＰスレッドを停止させ（ステップＳ１１４）、オブジェクト参照情報１４５０のメモリ領域の先頭位置に着目し（ステップＳ１１５）、参照処理部１４３０に参照処理を実行させ（ステップＳ１１６）、ＧＣフラグ１４６０をＯＦＦにし（ステップＳ１１７）、スレッド制御部３１０に全ＡＰスレッドの停止を解除させ（ステップＳ１１８）、解放部４４０にマーク付与がなされていないオブジェクトを解放させ（ステップＳ２２）、ＧＣ制御処理を終える。

ここで、インタプリタ部１１００によりなされる命令実行処理について簡単に説明する。

図１８は、実施形態２における命令実行処理を示すフローチャートである。

同図に示すように、この命令実行処理は、図１３で示した命令実行処理からス

テップＳ７６を削除したものと基本的に等しい。

まず、命令実行部１１０は、ＡＰスレッドの現在の実行位置における命令を解釈して実行し（ステップＳ７１）、オブジェクト参照検出部１１２０は、ＧＣフラグ１４６０がＯＮであるか否かを判定し（ステップＳ７２）、ＯＮでなければ、命令実行部１１０が次の命令の解釈実行を行う（ステップＳ７１）。

ステップＳ７２において、ＧＣフラグ３２５がＯＮであると判定した場合には、オブジェクト参照検出部１１２０は、ステップＳ７１において命令実行部１１０により実行された命令が、オブジェクトポインタを処理対象とする命令であったか否かを判定し（ステップＳ７３）、オブジェクトポインタを処理対象とする命令でなかったならば、命令実行部１１０が次の命令の解釈実行を行う（ステップＳ７１）。

ステップＳ７３において、オブジェクトポインタを処理対象とする命令であったと判定した場合には、オブジェクト参照検出部１１２０はそのオブジェクトポインタと同値のものがオブジェクト参照情報領域に既に格納済みであるか否かを判定し（ステップＳ７４）、格納済みである場合には、命令実行部１１０が次の命令の解釈実行を行う（ステップＳ７１）。

ステップＳ７４において、そのオブジェクトポインタと同値のものがオブジェクト参照情報のメモリ領域に格納済みでない場合には、オブジェクト参照検出部１１２０は、そのオブジェクトポインタを、オブジェクト参照情報中の位置に格納し（ステップＳ７５）、その後は命令実行部１１０が次の命令の解釈実行を行う（ステップＳ７１）。

このようにしてＧＣ制御部１４１０は、スレッド選定条件を参照することにより、参照処理の対象にするＡＰスレッドを１つ選定し、選択したＡＰスレッドについて、スレッド制御部３１０にそのＡＰスレッドを停止させてから、参照処理部１４３０にそのＡＰスレッドに対応するスタックに基づいて参照処理を行わせ、その後にスレッド制御部３１０に選択したＡＰスレッドの停止を解除させてから、次のＡＰスレッドの選択を行うという手順を、未処理のＡＰスレッドが存在しなくなるまで繰り返し、その後、スレッド制御部３１０に全ＡＰスレッドを停止させ、参照処理部１４３０にオブジェクト参照情報に基づいて参照処理を行わせて、スレッド制御部３１０に全ＡＰスレッドの停止を解除させて、解放部４４０に、

マーク付与がなされていないオブジェクトの解放を行わせる。

<3. 考察>

以下、上述の本発明に係るＧＣシステムが管理するオブジェクトと、従来のＪａｖａ実行環境におけるガーベジコレクタ等により扱われるオブジェクト(以下、特に「Ｊａｖａオブジェクト」という。）とのメモリ領域の差異について簡単に対比説明を行う。

上述の実施形態１、２に示した本発明に係るＧＣシステムでは、プログラマに確保したメモリの解放を意識させないようなＪａｖａ言語等のオブジェクト指向言語により作成されたＡＰを実行する際におけるオブジェクトの生成時のメモリ確保と、ＧＣ処理におけるそのオブジェクトに対応するメモリの解放とを、ＭＭＵを利用した従来のＯＳのメモリ管理機構を通じて行う。これにより、オブジェクトに対して確保されるメモリ領域は、Ｎａｔｉｖｅデータ、つまり他の言語、例えばＣ言語等、により作成されたプログラムのデータに対して確保されるメモリ領域と、論理アドレス空間において混在する。図３では、この混在するイメージを表現している。

これに対して、従来のＪａｖａ実行環境により管理されたＪａｖａオブジェクトと、ここではＣ言語のプログラムにおけるデータ（以下、「Ｃデータ」という。）とのメモリにおける存在状況を示した図が図１９である。

図１９に示すように、各Ｃ言語のプログラム９５３、９５４は、Ｃデータ９２１、９２２のためのメモリ領域をＯＳのメモリ管理機構を通じて、ランダムアクセスメモリ（ＲＡＭ）９００内に確保する。

また、各Ｊａｖａプログラム９５１、９５２から生成される各Ｊａｖａオブジェクト９１１～９１３は、１つのヒープ領域９１０内に確保され、そのヒープ領域９１０内での配置の管理は、ＯＳによってではなく、ガーベジコレクタ９５０を含むＪａｖａ実行環境によって行われる。このため、Ｊａｖａ実行環境は、まずＯＳのメモリ管理機構を通じてヒープ領域を確保しておいて、オブジェクトを生成する際には、そのヒープ領域内の一領域をそのオブジェクトに対して割り当てる。このヒープ領域は、他の言語のプログラムが利用できるメモリ量を減らしてしまう。

また、Ｊａｖａ実行環境は、オブジェクトを解放する場合には、ヒープ領域内

のそのオブジェクトに割り当てられていた領域を、新たにオブジェクトに対して割り当てられ得る未使用領域として管理するが、ヒープ領域内において断片化した全未使用領域を連続させるためのいわゆるメモリコンパクション処理を随時行う必要がある。

なお、この点、実施形態１、２に示した本発明に係るＧＣシステムでは、ヒープ領域の確保及びその内部の管理を行うのではなく、オブジェクトに対するメモリ割り当てを直接的に、ＭＭＵを利用した従来のＯＳのメモリ管理機構を通じて行っているため、当然ながらヒープ領域内のメモリコンパクション等は不要となる。

＜4.補足＞

以上、本発明に係るＧＣシステムについて、実施形態１、２に基づいて説明したが、本発明は、勿論これらの実施形態に限られない。以下、実施形態の変形に関して説明する。

（１）スレッド選定条件は、スレッド状態、スレッド優先度、スタックサイズという３つの条件をこの順に優先適用されることとしたが、条件数や適用順序等は、これに限定されることはなく、また、例えばスレッド優先度が高いものを条件にしたりスタックサイズが大きいものを条件にしたりしてもよい。

但し、実施形態において示したようにスレッド状態がｗａｉｔ状態のＡＰスレッドを参照処理対象と選定することは、現在動作しているＡＰに対する悪影響が少ないという効果につながる。また、参照処理（図１２）のアルゴリズムにより、あるＡＰスレッドに対する参照処理にかかる時間は、参照処理を先に行うＡＰスレッドより後に行うＡＰスレッドの方が比較的短くなるため、実施形態において示したようにスレッド優先度が低いものを早めに参照処理対象と選定するようにしておくことで、即応性の求められるところのスレッド優先度の高いＡＰスレッドの停止時間を短くし得るという効果が生じる。また、同様の理由と、スタックサイズが大きいほど参照処理には時間がかかるという理由とから、対応するスタックサイズの小さいＡＰスレッドから順に参照処理を行うようにすることで、同じスレッド優先度のＡＰスレッド間である程度等しくなるようにＡＰスレッドの停止時間を分散できるという効果が生じる。

（２）実施形態では、ＧＣ制御処理がタイマーに基づき一定周期で行われること

としたが、これに限定されることはなく、ＡＰに割り当てられる空きメモリの量が所定量より少なくなった場合にＧＣ制御処理を行うようにしてもよい。

（３）実施形態で示したオブジェクト参照情報のメモリ領域は、連続した物理アドレスで表されるメモリ領域であってもよいし、オブジェクトポインタの格納に際して必要が生じた度に一定量のメモリ領域を追加的に確保してその確保した複数の断続的なメモリ領域を論理アドレス上で連続しているように扱うこととしてもよい。

（４）実施形態１で示したＧＣシステムにおいては、インタプリタによる命令実行処理中に、ＧＣフラグがＯＮであればオブジェクトポインタを処理対象としたときにオブジェクト参照情報のメモリ領域にそのオブジェクトポインタを保存する処理を行っておくこととし、命令実行処理中では、そのオブジェクトポインタと同値のものをｆｒｏｍテーブルからｔｏテーブルに移行することを内容とする参照処理を行わずインタプリタによる命令実行の高速性を保つようにしたが、命令実行処理中にその参照処理を行うこととしてもよい。

（５）実施形態では共有オブジェクト管理情報内のオブジェクトポインタから辿ることのできるオブジェクトへのポインタは参照処理によってｆｒｏｍテーブルからｔｏテーブルに移行されることとしたが、参照処理を行わなくても、共有オブジェクト管理情報によって管理されているオブジェクトは解放部による解放対象から除外しさえすればよい。

（６）実施形態で示したＧＣシステムは、コンピュータ上に実装されるのみならず、ＣＰＵを備える携帯端末、家電機器等において実装されることとしてもよい。

（７）実施形態で示したスレッド制御部は、微小時間毎に各スレッドを切り替えて実行することにより各スレッドを並列的に実行、つまり擬似的に並列実行することとしたが、各スレッドをマルチプロセッサにおける複数のプロセッサエレメントによって実際に並列実行することとしてもよい。

（８）実施形態では、オブジェクト生成部により生成される各オブジェクトは、従来のＯＳのメモリ管理機構により、各Ｎａｔｉｖｅデータと同等に、論理アドレス空間にメモリが割り当てられるものであることとしたが、各オブジェクトは、論理アドレス空間におけるひとかたまりのヒープ領域内に、それぞれメモリが割り当てられるように、特別の管理を行うこととしてもよい。この特別の管理は、

ヒープ領域を、従来のＯＳのメモリ管理機構により確保し、オブジェクトの生成に対応してその内部の未割当領域を各オブジェクトに割り当て、オブジェクトの解放に対応してその内部の領域を未割当領域と定めることにより実現される。この特別の管理をする場合において、更に、ヒープ領域内において断片化した全未割当領域を連続させるためのいわゆるメモリコンパクション処理を随時行う必要がある。なお、このことから、ヒープ領域を利用する方式は、上述の<3. 考察>において示したように必ずしも最適なものではない。

（９）実施形態で示したＧＣシステムは従来のＯＳのメモリ管理機構を含んでいることとしたが、ＧＣシステムの外部に、そのＧＣシステムの基盤環境として、ＭＭＵを用いてメモリを管理する従来のＯＳのメモリ管理機構が存在することとし、ＧＣシステムはメモリの確保及び解放をその外部のメモリ管理機構を通じて行うこととしてもよい。

（１０）実施形態で示したＧＣシステムの各機能を実現させるための各種処理（図８～図１４、図１７、図１８参照）をＣＰＵに実行させるためのプログラムを、記録媒体に記録し又は各種通信路等を介して、流通させ頒布することもできる。このような記録媒体には、ＩＣカード、光ディスク、フレキシブルディスク、ＲＯＭ等がある。流通、頒布されたプログラムは、ＣＰＵを備える装置におけるＣＰＵで読み取り可能なメモリ等に格納されることにより利用に供され、そのＣＰＵがそのプログラムを実行することにより実施形態で示したＧＣシステムの各機能が実現される。

産業上の利用可能性

本発明に係るＧＣシステムは、アプリケーションプログラム（ＡＰ：Application Program）の作成者にメモリ領域の確保や解放を意識させないＪａｖａ等のオブジェクト指向のプログラミング言語で作成されたＡＰのためのコンピュータ上の実行環境として利用される。

請 求 の 範 囲

1．複数スレッドで構成されるオブジェクト指向プログラムの実行過程において不要になったオブジェクトに対応するメモリ領域を解放するガーベジコレクションシステムであって、

複数スレッドそれぞれを順に選択する選択手段と、

選択されたスレッドについて、当該スレッドの実行を停止して、当該スレッドからオブジェクトポインタの参照を通じてアクセス可能であるオブジェクトを検出して当該検出したオブジェクトを非解放対象として管理して、当該スレッドの実行を再開するという手順からなる検査処理を実施する検査手段と、

前記選択手段による前記選択が開始された後において、実行中のスレッドによりオブジェクトポインタが処理対象にされたことを検知した場合には、当該オブジェクトポインタが指すオブジェクトを非解放対象として管理する検知手段と、

前記複数スレッド全てについて前記検査処理が完了した後において、非解放対象として管理されているオブジェクト以外のオブジェクトに対応するメモリ領域を、解放する解放手段とを備える

ことを特徴とするガーベジコレクションシステム。

2．前記検知手段は、前記検知を、実行中のスレッドについて未だ検査処理がなされていない場合に限り行い、

前記検知手段は、

前記検知をした場合に、当該実行中のスレッドに処理対象にされたオブジェクトポインタ、及び当該オブジェクトポインタから辿ることのできるオブジェクト内のオブジェクトポインタを、当該スレッドに対応する作業用メモリ領域に保存する検出部と、

前記検査手段によりスレッドの実行が停止されている間に、当該スレッドに対応する作業用メモリ領域内のオブジェクトポインタから辿ることのできるオブジェクトを非解放対象として管理する管理部とを有する

ことを特徴とする請求の範囲第１項記載のガーベジコレクションシステム。

3．前記検査処理は、

選択された当該スレッドに対応するスタック内のオブジェクトポインタが指す

オブジェクトを、前記アクセス可能であるとして検出した際において、

検出した当該オブジェクトが既に非解放対象として管理されておらず、かつ、当該オブジェクト内にオブジェクトポインタがあるときに限り、当該オブジェクトポインタが指すオブジェクトを更に前記アクセス可能であるとして検出する手順を、

繰り返し行うことを内容とする処理であり、

前記選択手段は、最初の選択後は、前記検査手段により前記検査処理が行われた後において前記複数スレッドのうち前記検査処理を実施されていないスレッドがある限り更なる選択を行い、

前記選択手段は、各スレッドに関する情報を参照し、所定のスレッド選定条件に基づき、前記選択を行う

ことを特徴とする請求の範囲第２項記載のガーベジコレクションシステム。

４．前記スレッド選定条件は、スレッド状態がｗａｉｔ状態であるスレッドをスレッド状態がｗａｉｔ状態以外であるスレッドよりも早期に選択することを示す条件を含み、

前記選択手段は、前記選択を行う時点でｗａｉｔ状態のスレッドがある限り、当該ｗａｉｔ状態のスレッドを選択する

ことを特徴とする請求の範囲第３項記載のガーベジコレクションシステム。

５．前記スレッド選定条件は、スレッド優先度のスレッド優先度の低いスレッドをスレッド優先度の高いスレッドよりも早期に選択することを示す条件を含む

ことを特徴とする請求の範囲第４項記載のガーベジコレクションシステム。

６．前記スレッド選定条件は、スレッドに対応するスタックサイズが小さいスレッドをスタックサイズが大きいスレッドよりも早期に選択することを示す条件を含む

ことを特徴とする請求の範囲第５項記載のガーベジコレクションシステム。

７．前記ガーベジコレクションシステムは、メモリ管理ユニット（ＭＭＵ）を用いてメモリを管理するメモリ管理機構を備え、オブジェクトを生成する必要がある度に、前記メモリ管理機構により当該オブジェクトに対応するメモリ領域を確保し、

前記解放手段は、前記メモリ管理機構を介して、メモリ領域の解放を行う

ことを特徴とする請求の範囲第６項記載のガーベジコレクションシステム。

８．前記スレッド選定条件は、スレッドに対応するスタックサイズが小さいスレッドをスタックサイズが大きいスレッドよりも早期に選択することを示す条件を含む

ことを特徴とする請求の範囲第３項記載のガーベジコレクションシステム。

９．前記スレッド選定条件は、スレッド優先度のスレッド優先度の低いスレッドをスレッド優先度の高いスレッドよりも早期に選択することを示す条件を含む

ことを特徴とする請求の範囲第３項記載のガーベジコレクションシステム。

１０．前記ガーベジコレクションシステムは、メモリ管理ユニット（MMU）を用いてメモリを管理するメモリ管理機構を利用するものであり、オブジェクトを生成する必要がある度に、前記メモリ管理機構により当該オブジェクトに対応するメモリ領域を確保し、

前記解放手段は、前記メモリ管理機構を介して、メモリ領域の解放を行う

ことを特徴とする請求の範囲第１項記載のガーベジコレクションシステム。

１１．コンピュータ上において複数スレッドで構成されるオブジェクト指向プログラムの実行過程において不要になったオブジェクトに対応するメモリ領域を解放するガーベジコレクション方法であって、

複数スレッドそれぞれを順に選択する選択ステップと、

選択されたスレッドについて、当該スレッドの実行を停止して、当該スレッドからオブジェクトポインタの参照を通じてアクセス可能であるオブジェクトを検出して当該検出したオブジェクトを非解放対象として管理して、当該スレッドの実行を再開するという手順からなる検査処理を実施する検査ステップと、

前記選択ステップによる前記選択が開始された後において、実行中のスレッドによりオブジェクトポインタが処理対象にされたことを検知した場合には、当該オブジェクトポインタが指すオブジェクトを非解放対象として管理する検知ステップと、

前記複数スレッド全てについて前記検査処理が完了した後において、非解放対象として管理されているオブジェクト以外のオブジェクトに対応するメモリ領域を、解放する解放ステップとを含む

ことを特徴とするガーベジコレクション方法。

１２．前記コンピュータは、メモリ管理ユニット（ＭＭＵ）を用いてメモリを管理するメモリ管理機構を利用するものであって、オブジェクト指向プログラムの実行過程においてオブジェクトを生成する必要がある度に、前記メモリ管理機構により当該オブジェクトに対応するメモリ領域を確保するものであり、

前記解放ステップは、前記メモリ管理機構を介して、メモリ領域の解放を行うことを特徴とする請求の範囲第１１項記載のガーベジコレクション方法。

１３．複数スレッドで構成されるオブジェクト指向プログラムの実行過程において不要になったオブジェクトに対応するメモリ領域を解放するガーベジコレクション処理をコンピュータに実行させるためのコンピュータプログラムであって、

前記ガーベジコレクション処理は、

複数スレッドそれぞれを順に選択する選択ステップと、

選択されたスレッドについて、当該スレッドの実行を停止して、当該スレッドからオブジェクトポインタの参照を通じてアクセス可能であるオブジェクトを検出して当該検出したオブジェクトを非解放対象として管理して、当該スレッドの実行を再開するという手順からなる検査処理を実施する検査ステップと、

前記選択ステップによる前記選択が開始された後において、実行中のスレッドによりオブジェクトポインタが処理対象にされたことを検知した場合には、当該オブジェクトポインタが指すオブジェクトを非解放対象として管理する検知ステップと、

前記複数スレッド全てについて前記検査処理が完了した後において、非解放対象として管理されているオブジェクト以外のオブジェクトに対応するメモリ領域を、解放する解放ステップとを含む

ことを特徴とするコンピュータプログラム。

１４．前記コンピュータは、メモリ管理ユニット（ＭＭＵ）を用いてメモリを管理するメモリ管理機構を利用するものであって、オブジェクト指向プログラムの実行過程においてオブジェクトを生成する必要がある度に、前記メモリ管理機構により当該オブジェクトに対応するメモリ領域を確保するものであり、

前記解放ステップは、前記メモリ管理機構を介して、メモリ領域の解放を行うことを特徴とする請求の範囲第１３項記載のコンピュータプログラム。

図1

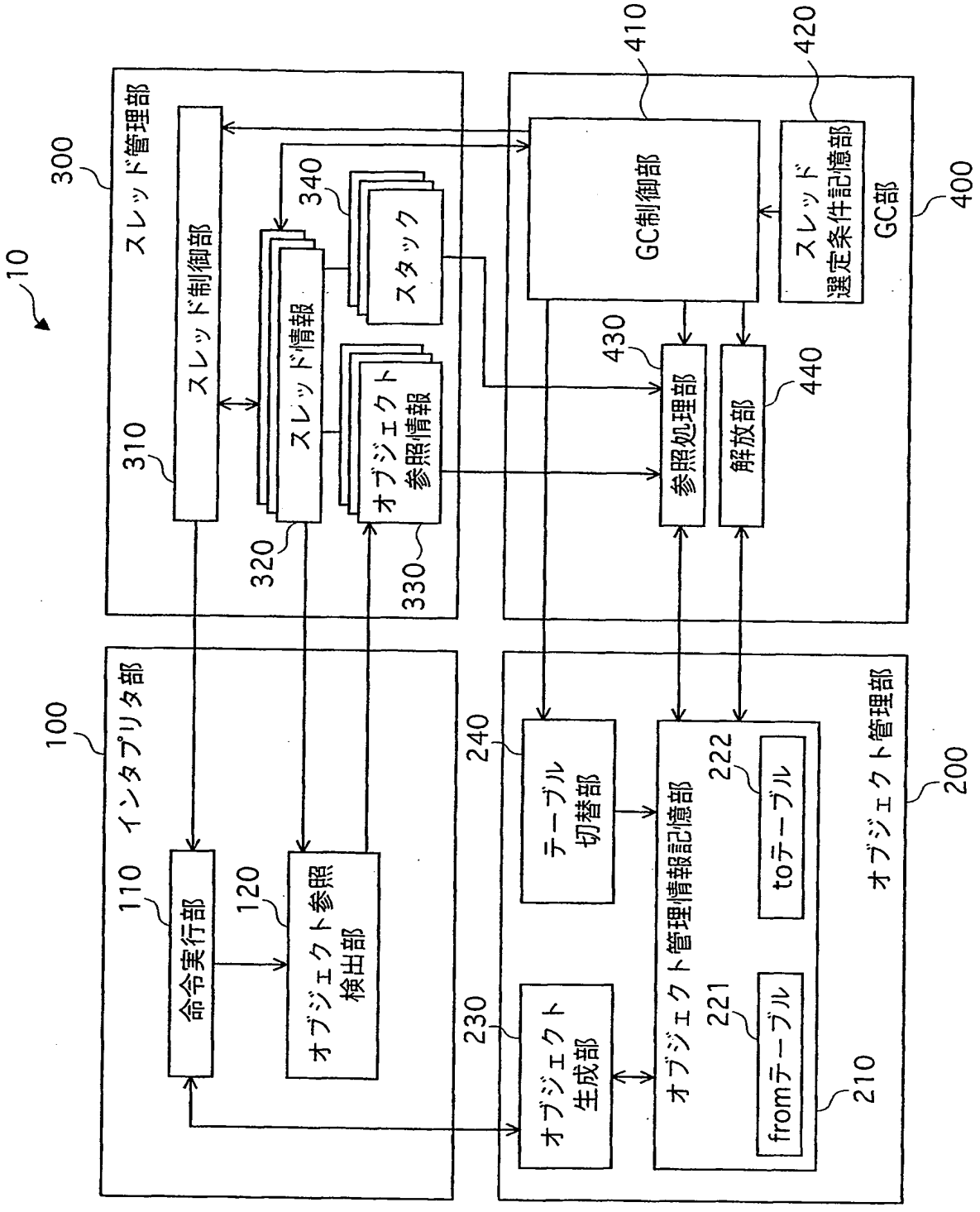

10
スレッド管理部
300
スレッド制御部
310
スレッド情報
320
スタック
340
オブジェクト参照情報
330
GC制御部
410
スレッド選定条件記憶部
420
GC部
400
参照処理部
430
解放部
440
インタプリタ部
100
命令実行部
110
オブジェクト参照検出部
120
オブジェクト管理部
200
テーブル切替部
240
オブジェクト管理情報記憶部
222
toテーブル
fromテーブル
221
210
オブジェクト生成部
230

図2

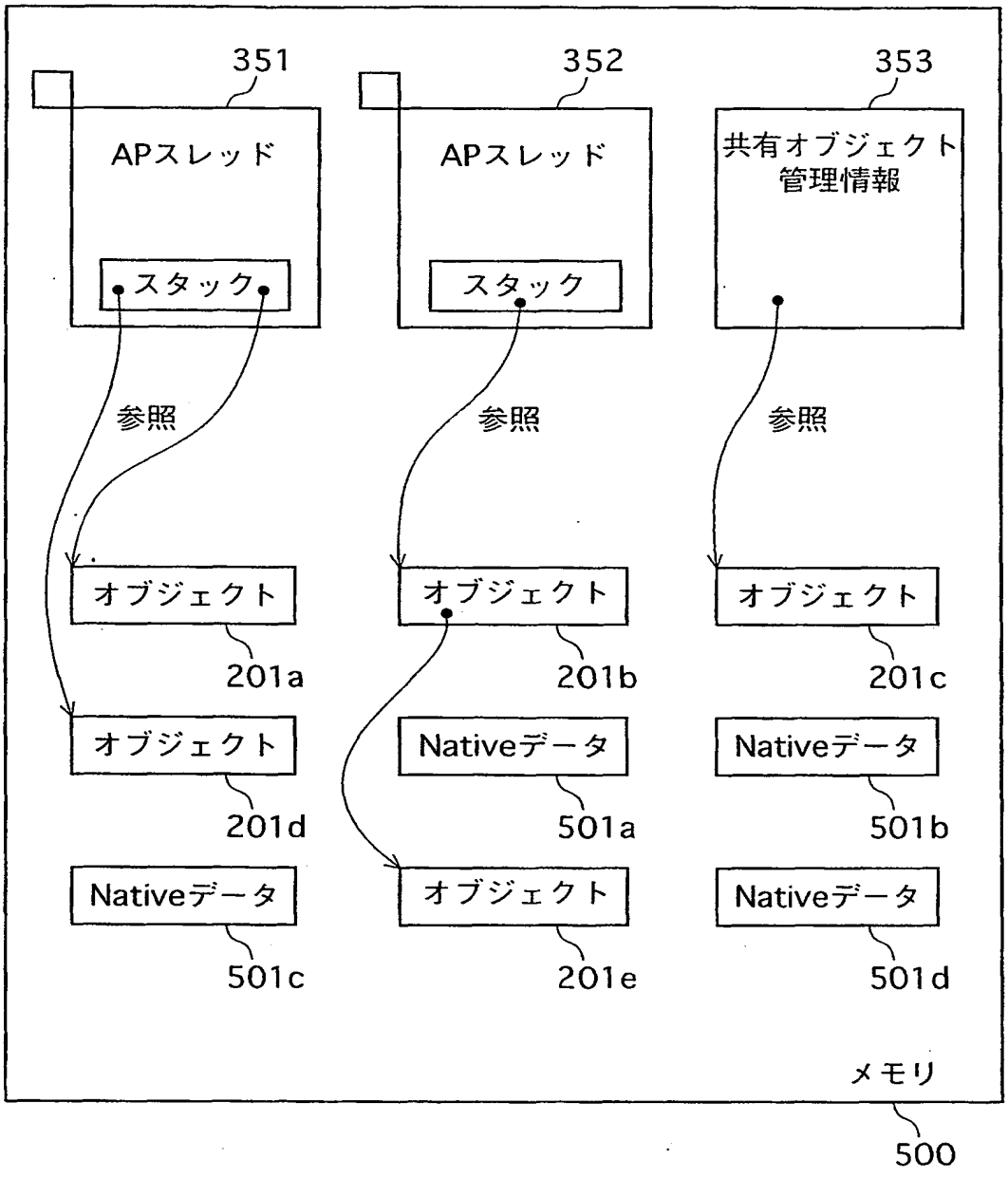

351
352
353
APスレッド
APスレッド
共有オブジェクト
管理情報
スタック
スタック
参照
参照
参照
オブジェクト
オブジェクト
オブジェクト
201a
201b
201c
オブジェクト
Nativeデータ
Nativeデータ
201d
501a
501b
Nativeデータ
オブジェクト
Nativeデータ
501c
201e
501d
メモリ
500

図3

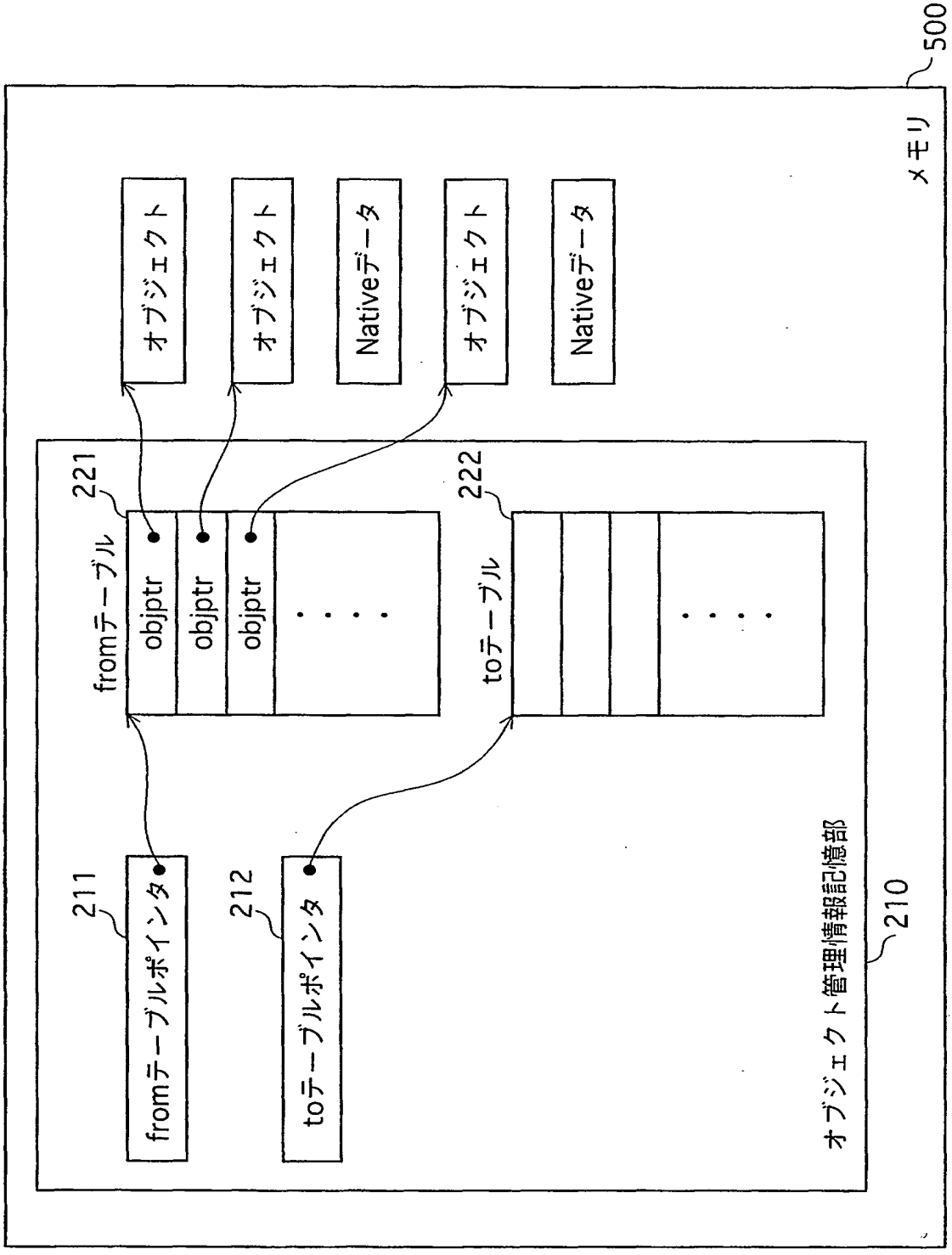

500
メモリ
オブジェクト
オブジェクト
Nativeデータ
オブジェクト
Nativeデータ
221
222
fromテーブル
objptr
objptr
objptr
toテーブル
211
212
fromテーブルポインタ
toテーブルポインタ
オブジェクト管理情報記憶部
210

図4A

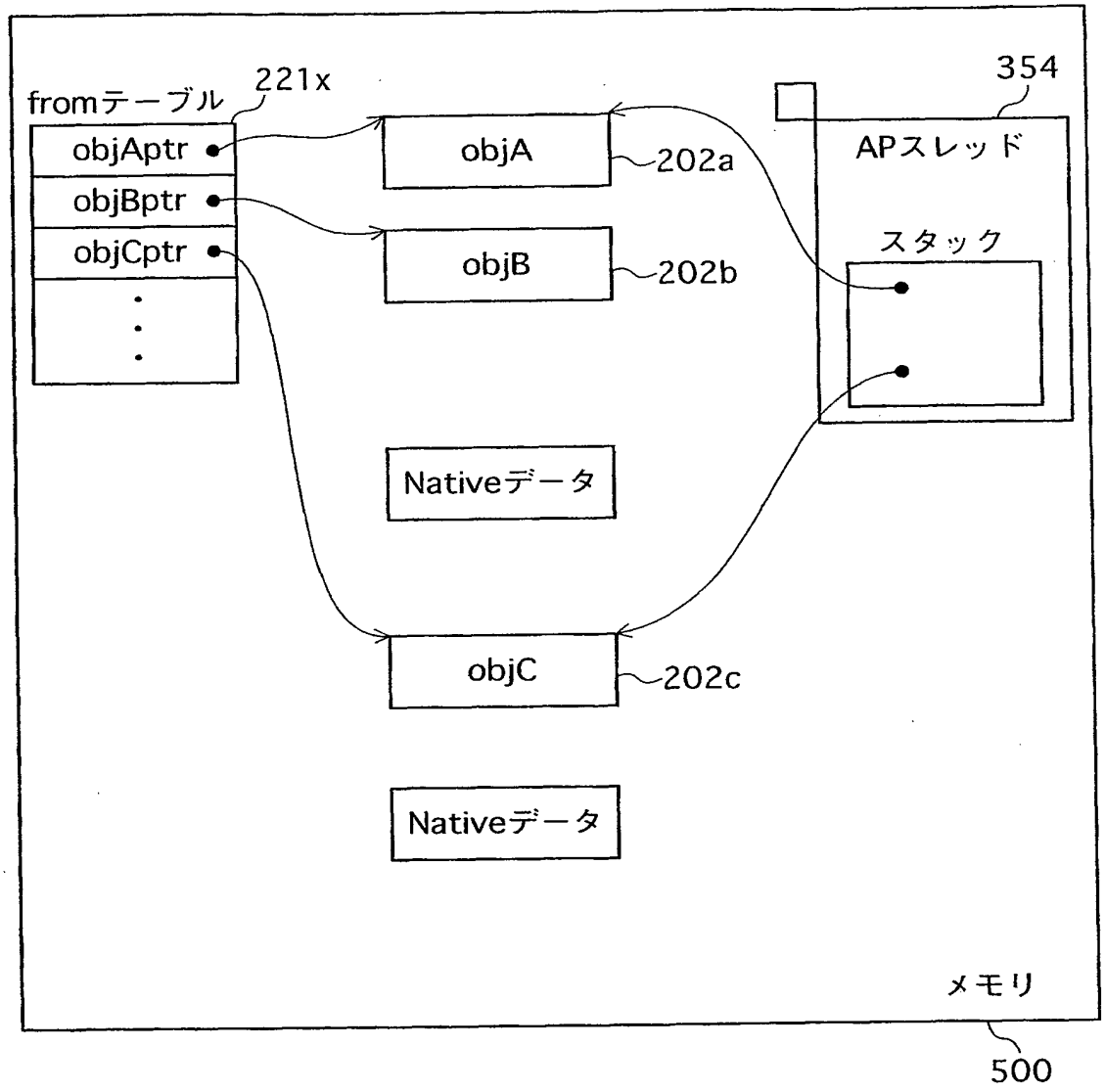

fromテーブル
221x
objAptr
objBptr
objCptr
objA
202a
objB
202b
354
APスレッド
スタック
Nativeデータ
objC
202c
Nativeデータ
メモリ
500

図4B

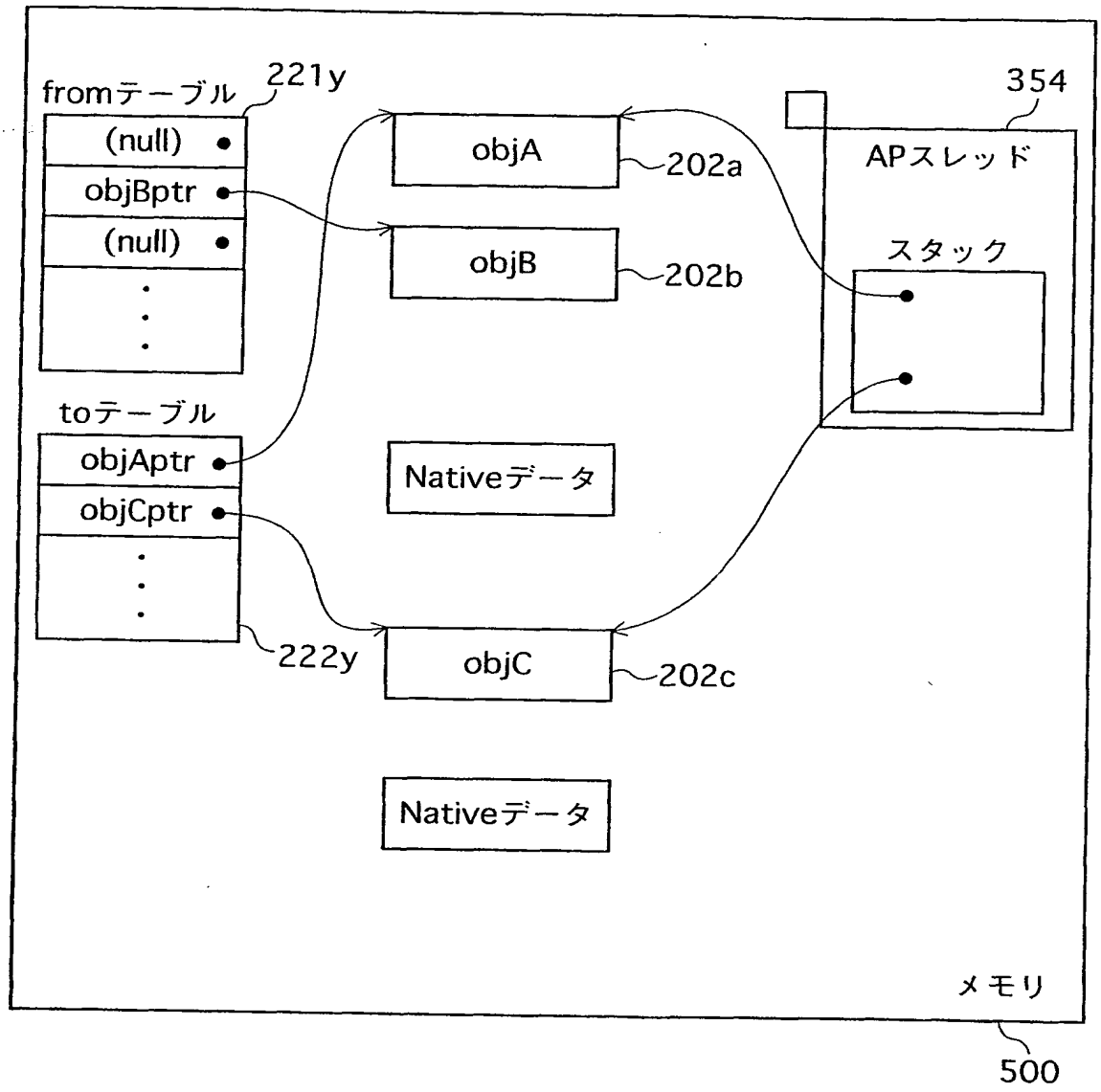

fromテーブル
221y
(null)
objBptr
(null)
toテーブル
objAptr
objCptr
222y
objA
202a
objB
202b
objC
202c
Nativeデータ
Nativeデータ
354
APスレッド
スタック
メモリ
500

図5

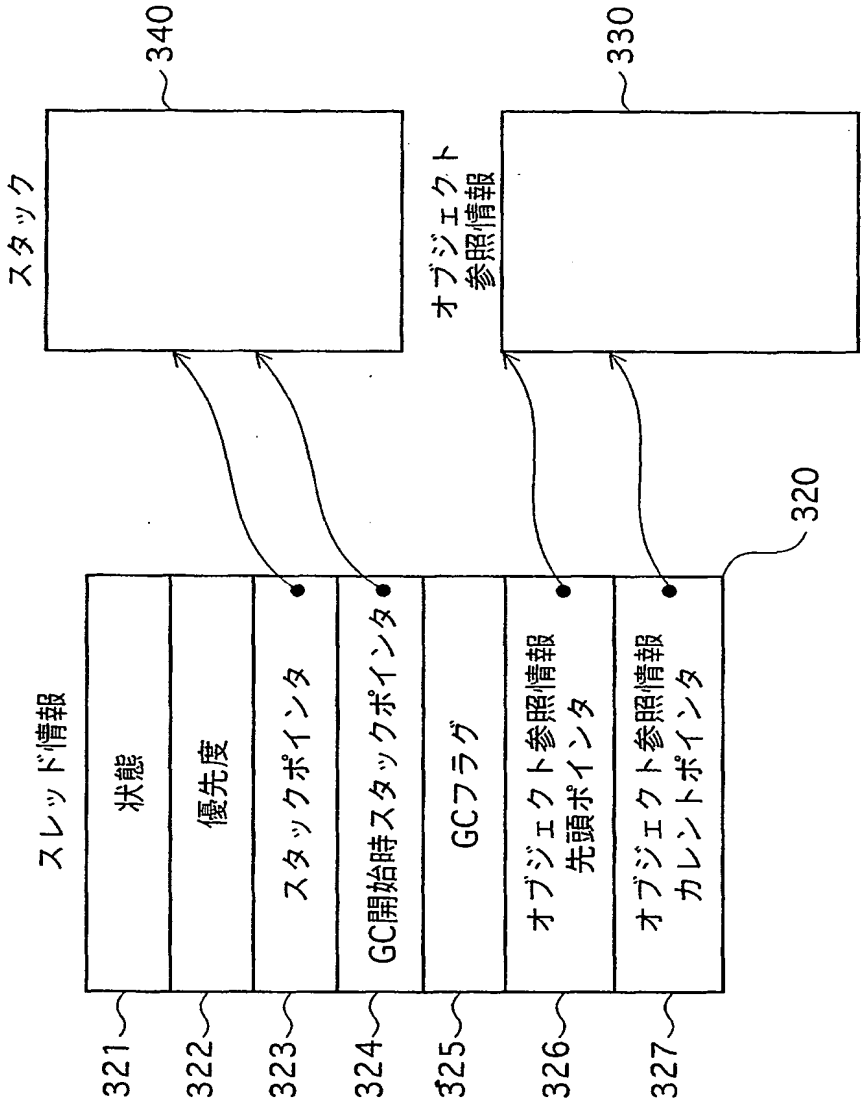

スタック
340
オブジェクト
参照情報
330
スレッド情報
状態
優先度
スタックポインタ
GC開始時スタックポインタ
GCフラグ
オブジェクト参照情報
先頭ポインタ
オブジェクト参照情報
カレントポインタ
320
321
322
323
324
325
326
327

図6

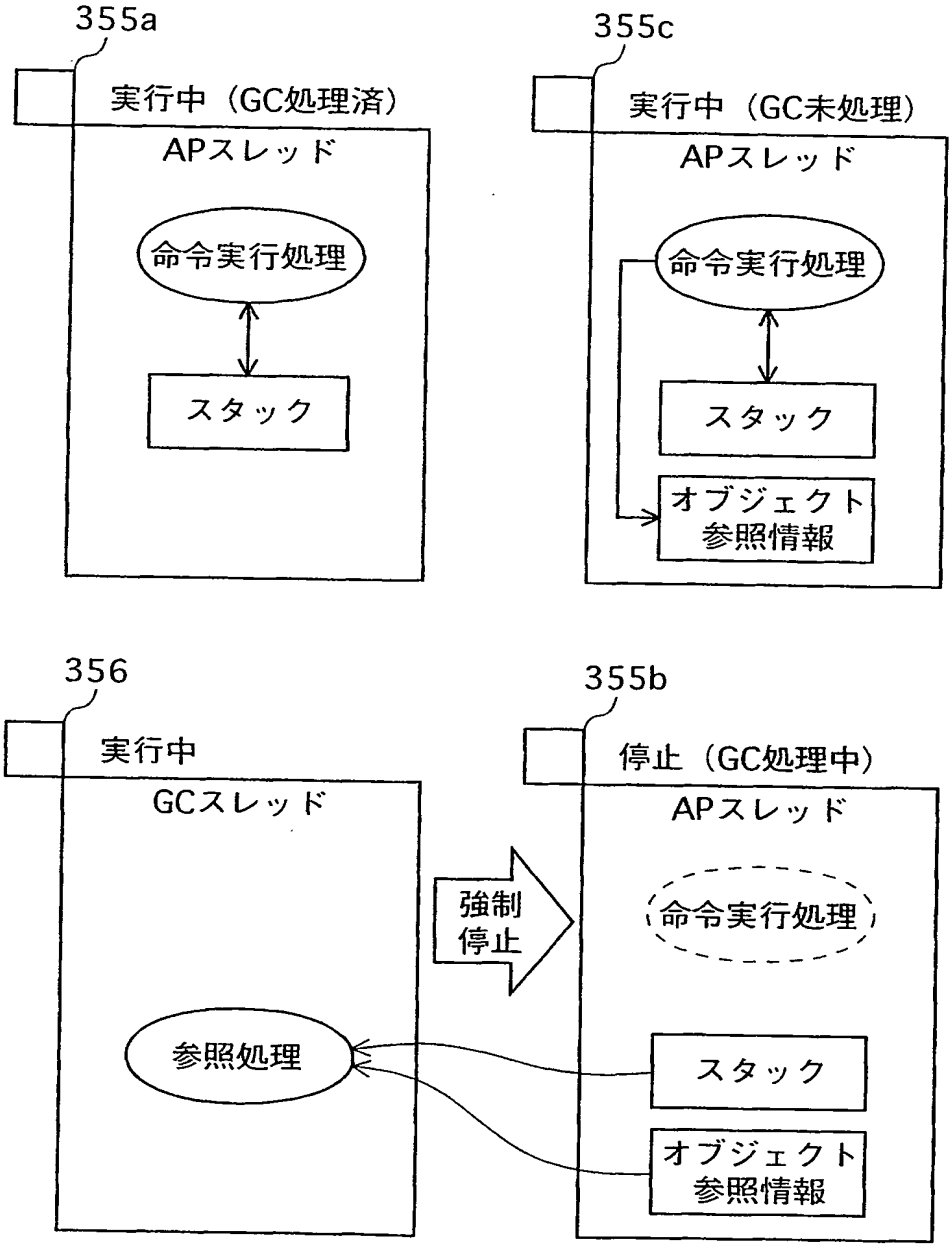

355a
実行中（GC処理済）
APスレッド
命令実行処理
スタック
355c
実行中（GC未処理）
APスレッド
命令実行処理
スタック
オブジェクト参照情報
356
実行中
GCスレッド
参照処理
強制停止
355b
停止（GC処理中）
APスレッド
命令実行処理
スタック
オブジェクト参照情報

図7

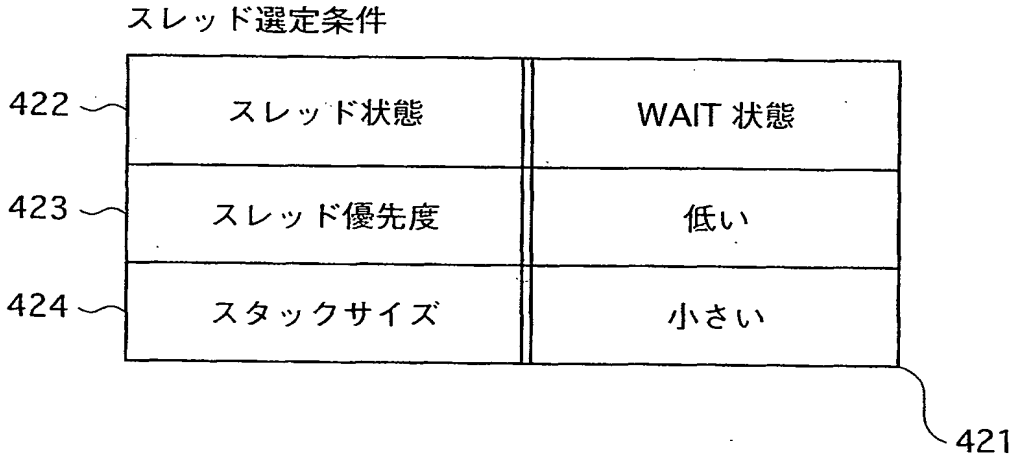

スレッド選定条件
422
スレッド状態
WAIT 状態
423
スレッド優先度
低い
424
スタックサイズ
小さい
421

図8

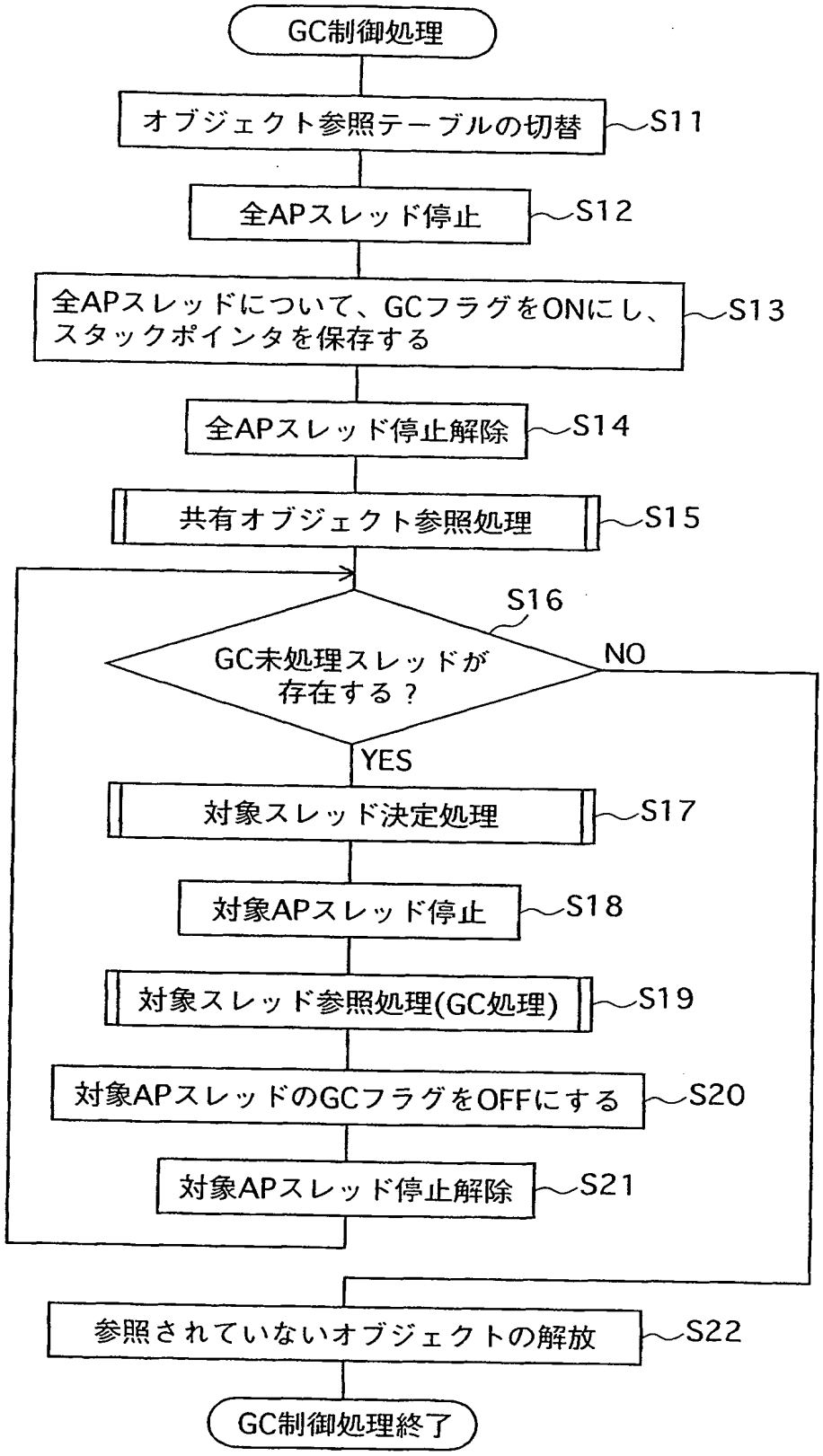

GC制御処理
オブジェクト参照テーブルの切替
S11
全APスレッド停止
S12
全APスレッドについて、GCフラグをONにし、スタックポインタを保存する
S13
全APスレッド停止解除
S14
共有オブジェクト参照処理
S15
S16
GC未処理スレッドが存在する？
NO
YES
対象スレッド決定処理
S17
対象APスレッド停止
S18
対象スレッド参照処理(GC処理)
S19
対象APスレッドのGCフラグをOFFにする
S20
対象APスレッド停止解除
S21
参照されていないオブジェクトの解放
S22
GC制御処理終了

図9

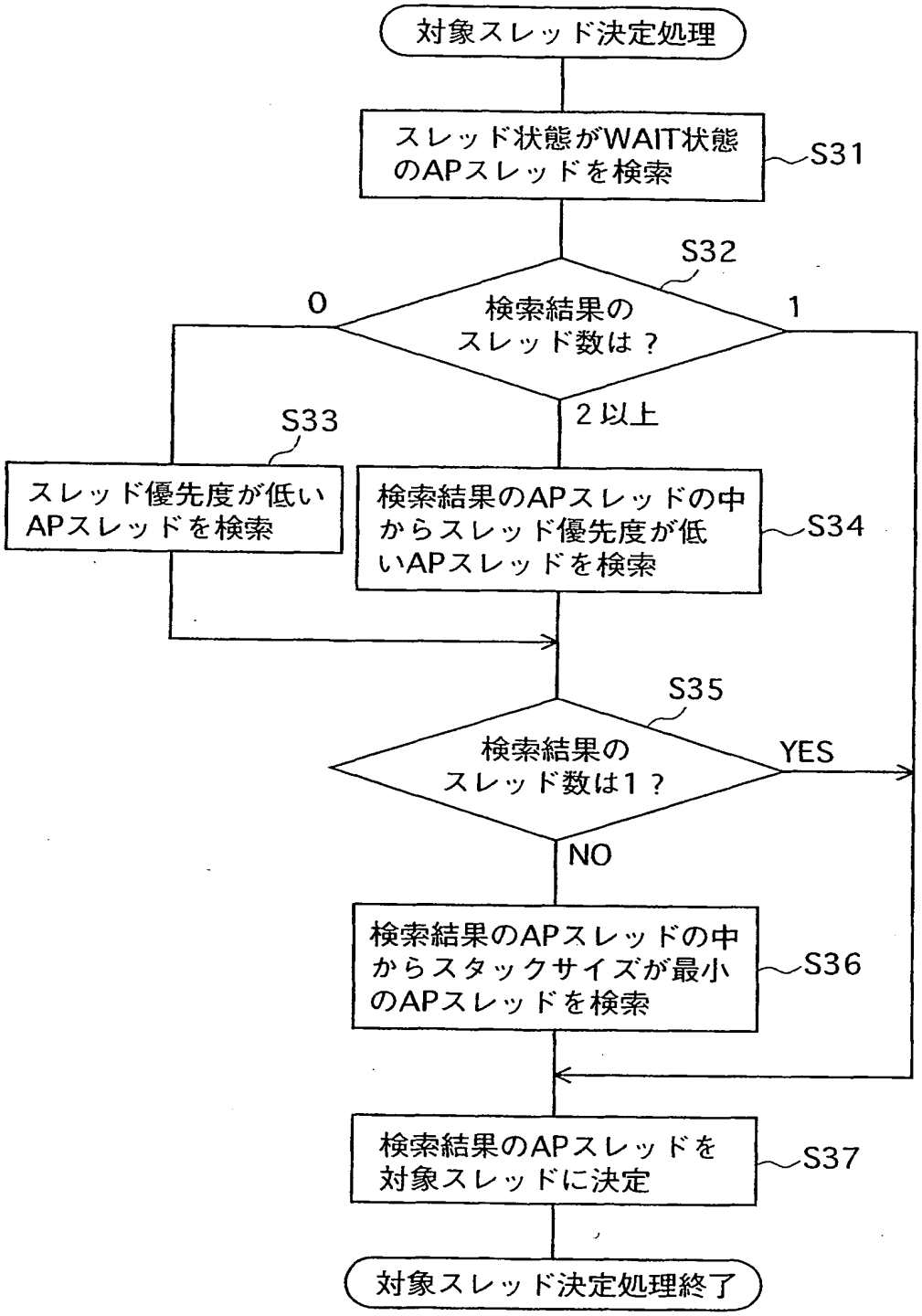

対象スレッド決定処理
スレッド状態がWAIT状態のAPスレッドを検索
S31
S32
検索結果のスレッド数は？
0
1
2以上
S33
スレッド優先度が低いAPスレッドを検索
検索結果のAPスレッドの中からスレッド優先度が低いAPスレッドを検索
S34
S35
検索結果のスレッド数は1？
YES
NO
検索結果のAPスレッドの中からスタックサイズが最小のAPスレッドを検索
S36
検索結果のAPスレッドを対象スレッドに決定
S37
対象スレッド決定処理終了

図10

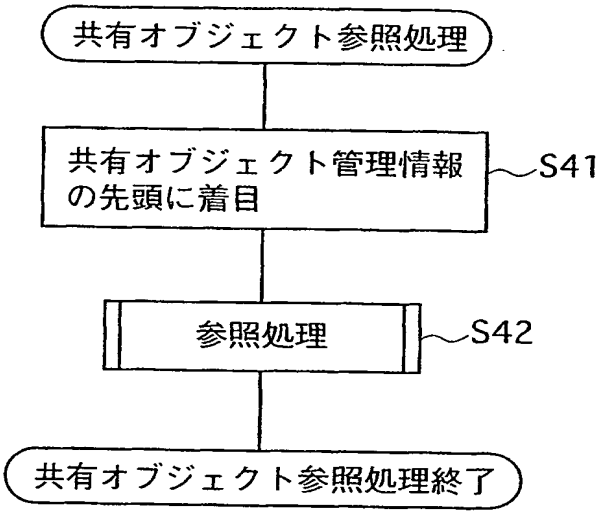

共有オブジェクト参照処理
共有オブジェクト管理情報の先頭に着目
S41
参照処理
S42
共有オブジェクト参照処理終了

図11

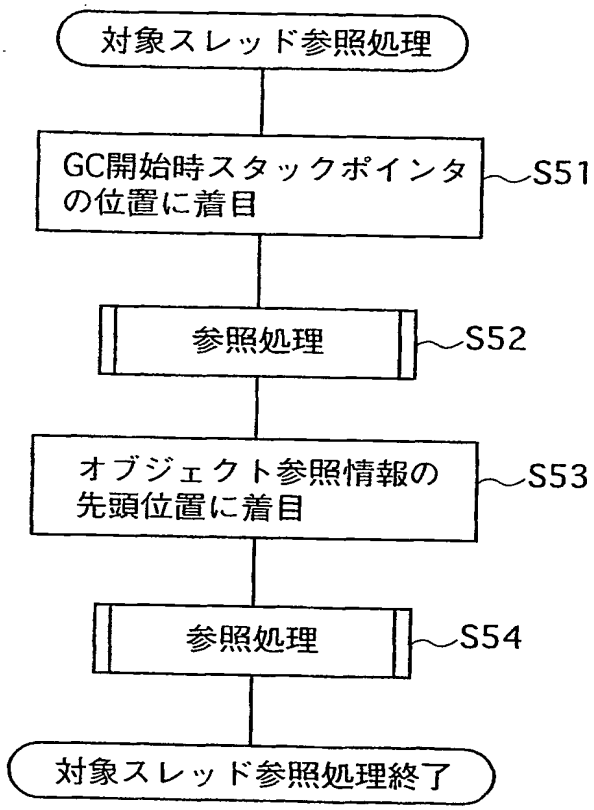

対象スレッド参照処理
GC開始時スタックポインタの位置に着目
S51
参照処理
S52
オブジェクト参照情報の先頭位置に着目
S53
参照処理
S54
対象スレッド参照処理終了

図12

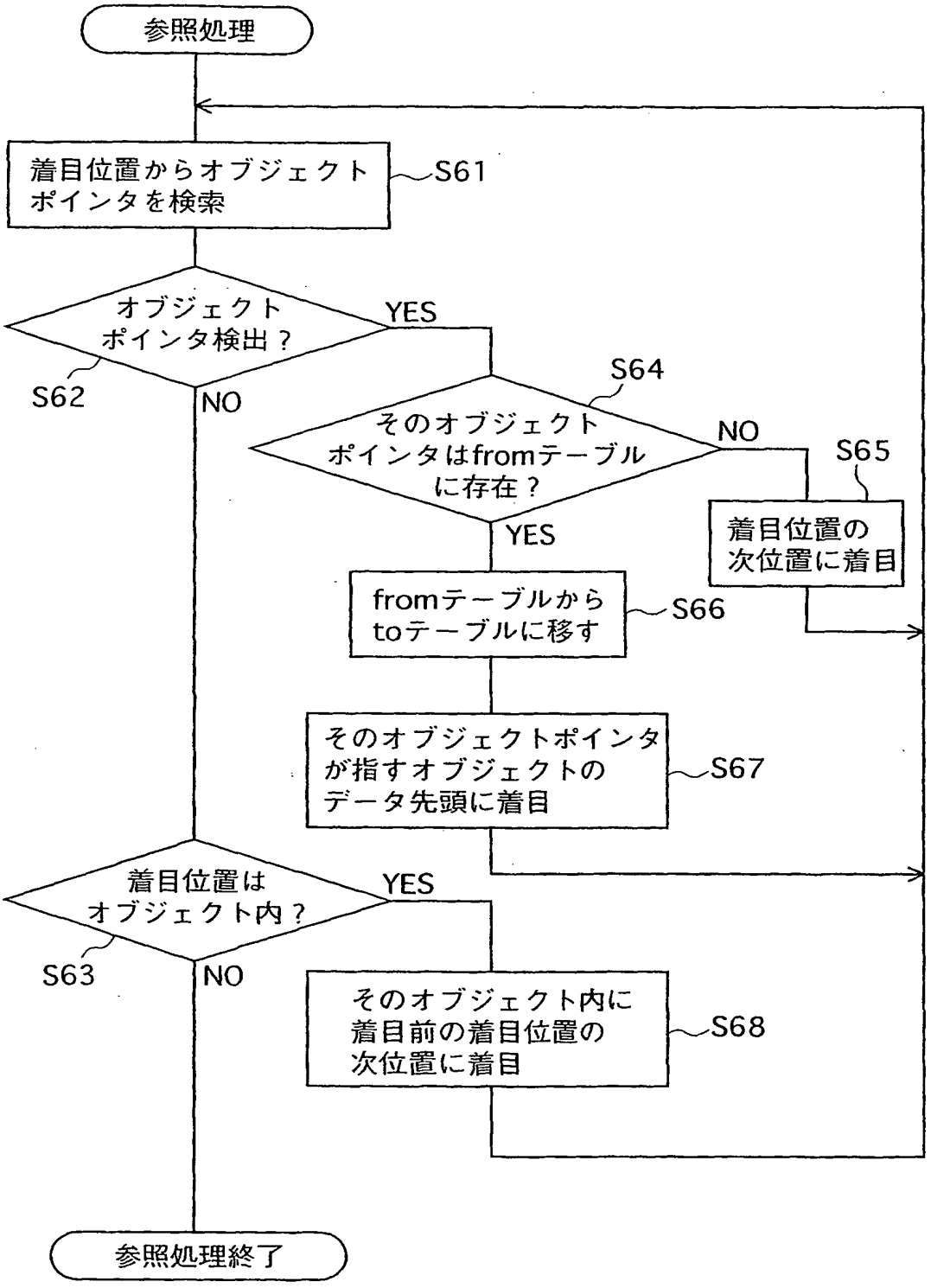

参照処理
着目位置からオブジェクトポインタを検索
S61
オブジェクトポインタ検出？
YES
NO
S62
S64
そのオブジェクトポインタはfromテーブルに存在？
NO
YES
S65
着目位置の次位置に着目
fromテーブルからtoテーブルに移す
S66
そのオブジェクトポインタが指すオブジェクトのデータ先頭に着目
S67
着目位置はオブジェクト内？
YES
NO
S63
そのオブジェクト内に着目前の着目位置の次位置に着目
S68
参照処理終了

図13

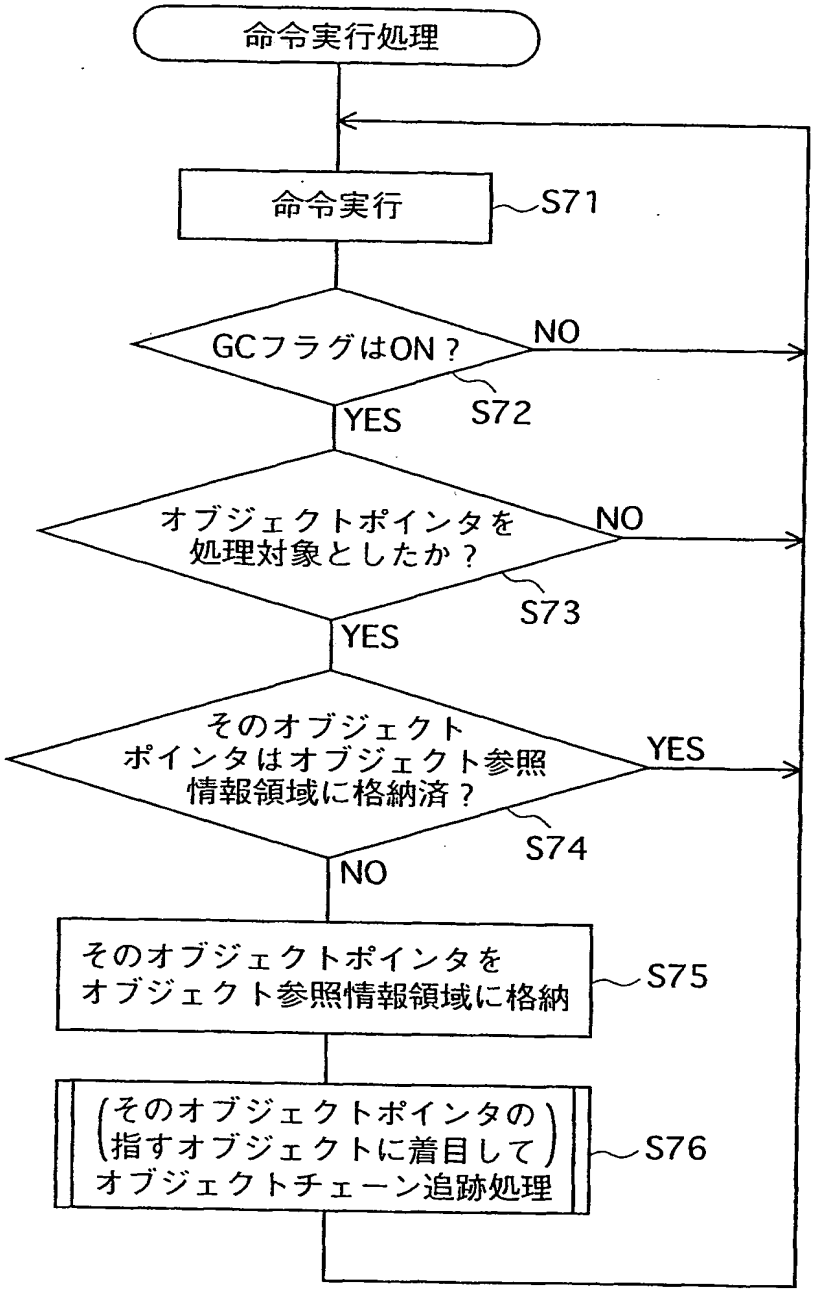

命令実行処理
命令実行
S71
GCフラグはON？
NO
S72
YES
オブジェクトポインタを
処理対象としたか？
NO
S73
YES
そのオブジェクト
ポインタはオブジェクト参照
情報領域に格納済？
YES
S74
NO
そのオブジェクトポインタを
オブジェクト参照情報領域に格納
S75
(そのオブジェクトポインタの
指すオブジェクトに着目して)
オブジェクトチェーン追跡処理
S76

図14

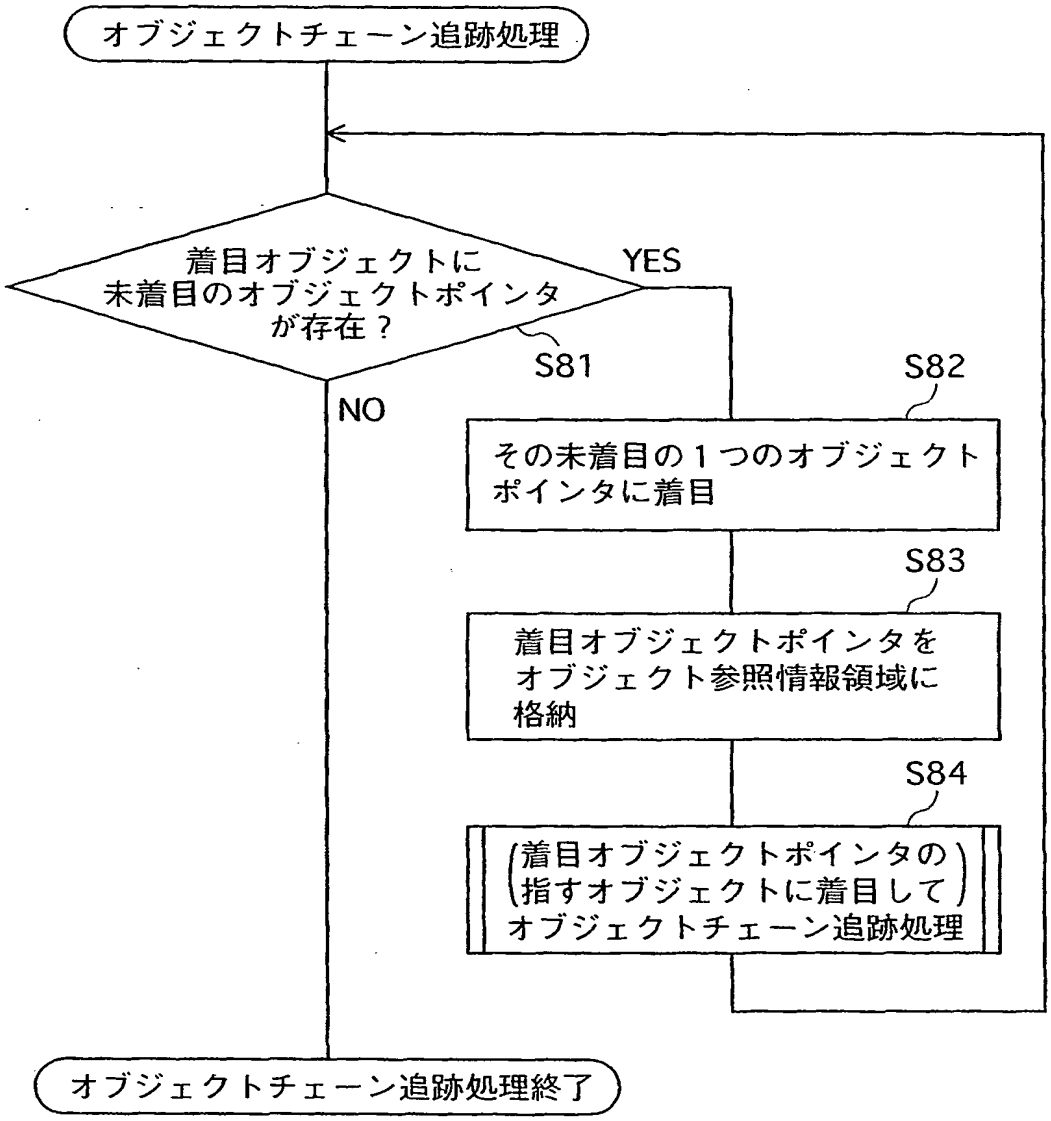

オブジェクトチェーン追跡処理
着目オブジェクトに
未着目のオブジェクトポインタ
が存在？
YES
NO
S81
S82
その未着目の１つのオブジェクト
ポインタに着目
S83
着目オブジェクトポインタを
オブジェクト参照情報領域に
格納
S84
着目オブジェクトポインタの
指すオブジェクトに着目して
オブジェクトチェーン追跡処理
オブジェクトチェーン追跡処理終了

図15

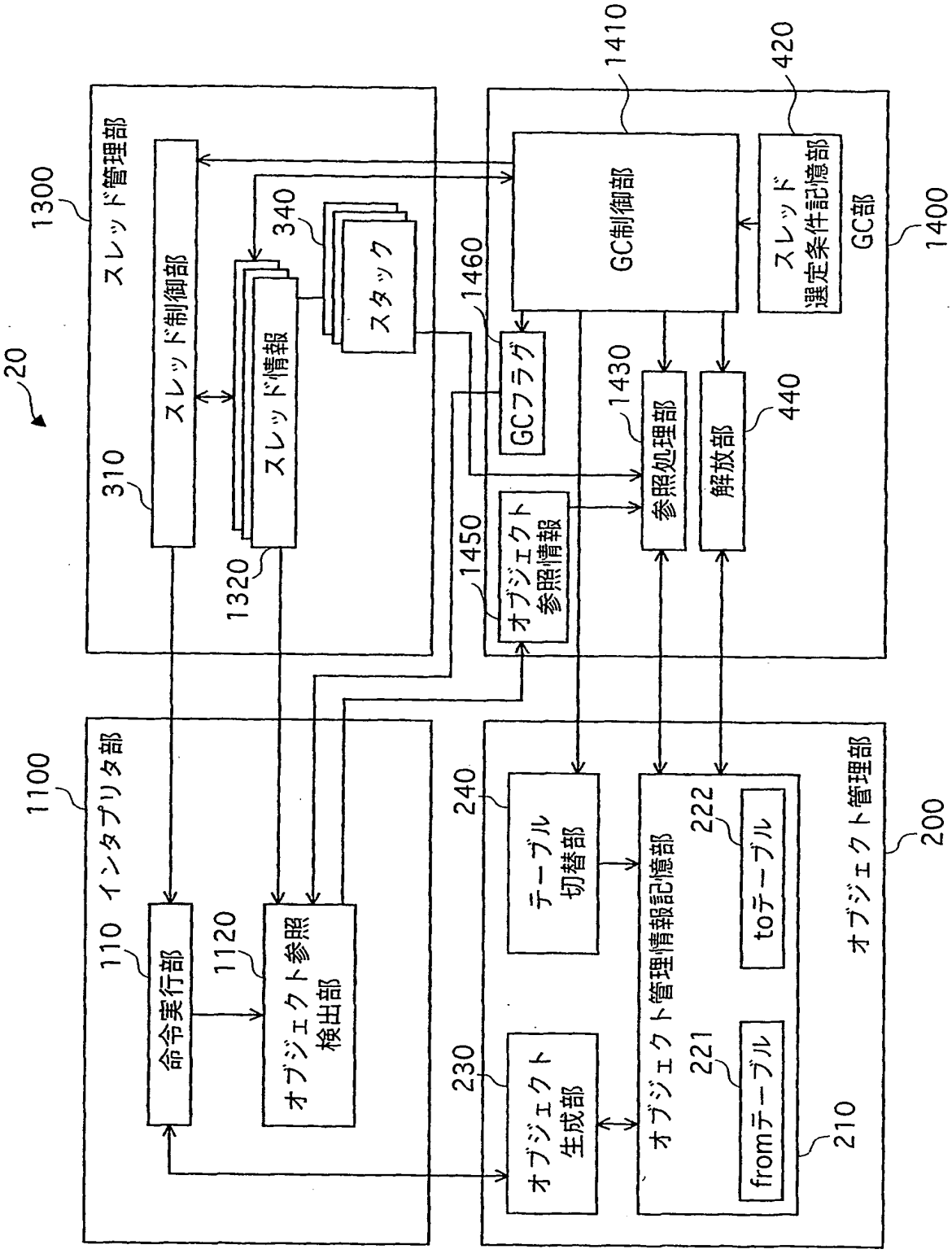

20
1300
スレッド管理部
スレッド制御部
310
スレッド情報
1320
340
スタック
1460
GCフラグ
1410
GC制御部
420
スレッド選定条件記憶部
GC部
1400
1430
参照処理部
440
解放部
1450
オブジェクト参照情報
1100
インタプリタ部
110
命令実行部
1120
オブジェクト参照検出部
240
テーブル切替部
222
toテーブル
オブジェクト管理情報記憶部
221
fromテーブル
230
オブジェクト生成部
210
オブジェクト管理部
200

図16

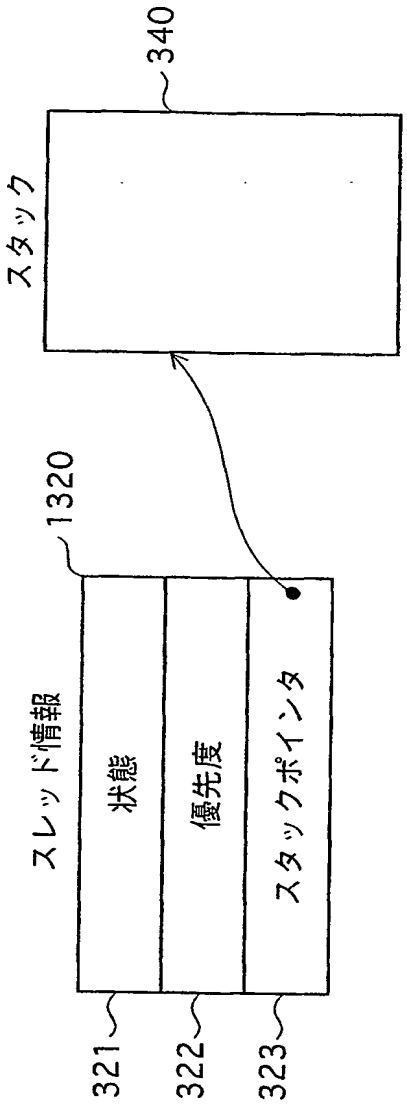


図17

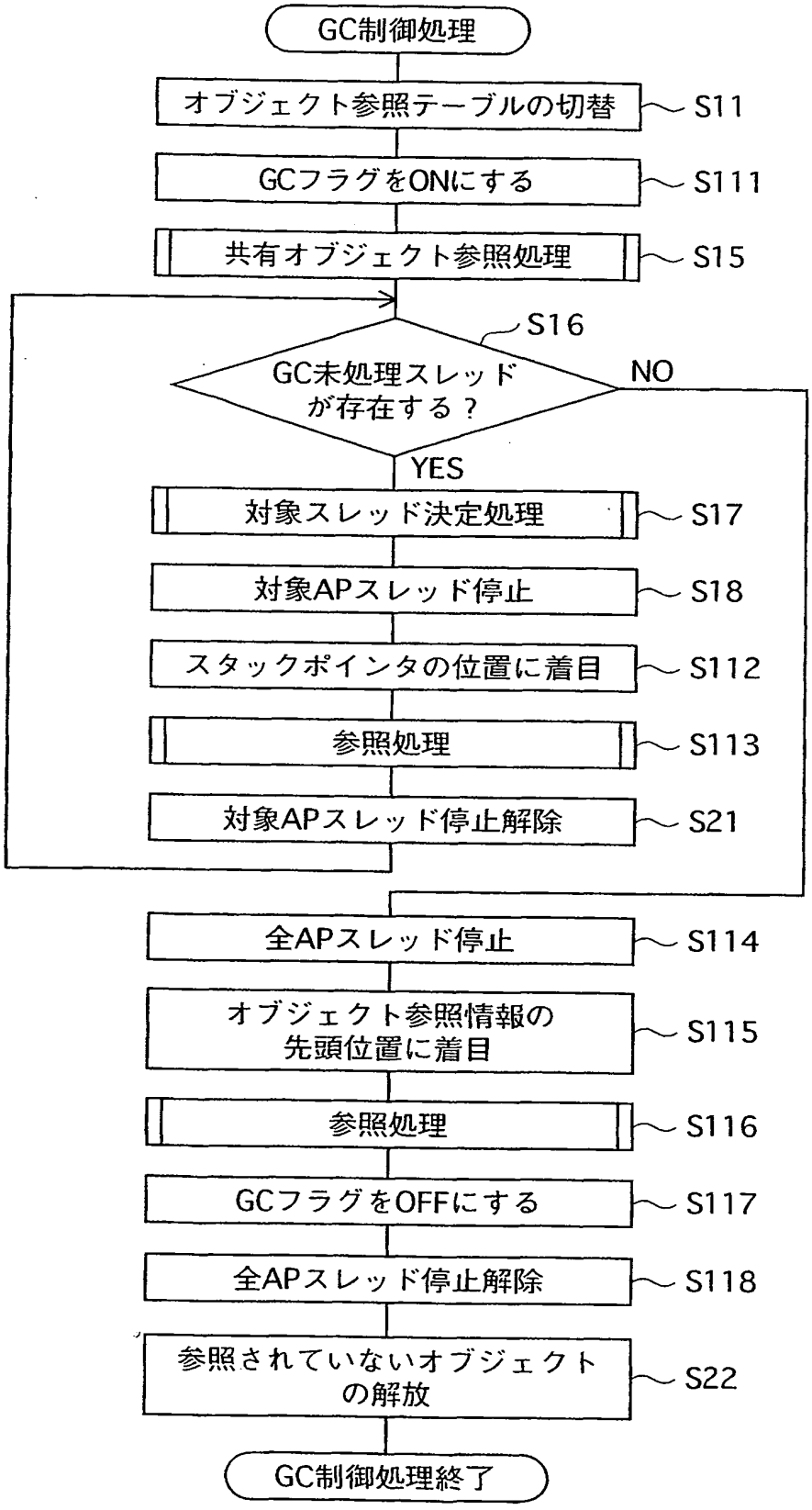

GC制御処理
オブジェクト参照テーブルの切替
S11
GCフラグをONにする
S111
共有オブジェクト参照処理
S15
S16
GC未処理スレッドが存在する？
NO
YES
対象スレッド決定処理
S17
対象APスレッド停止
S18
スタックポインタの位置に着目
S112
参照処理
S113
対象APスレッド停止解除
S21
全APスレッド停止
S114
オブジェクト参照情報の先頭位置に着目
S115
参照処理
S116
GCフラグをOFFにする
S117
全APスレッド停止解除
S118
参照されていないオブジェクトの解放
S22
GC制御処理終了

図18

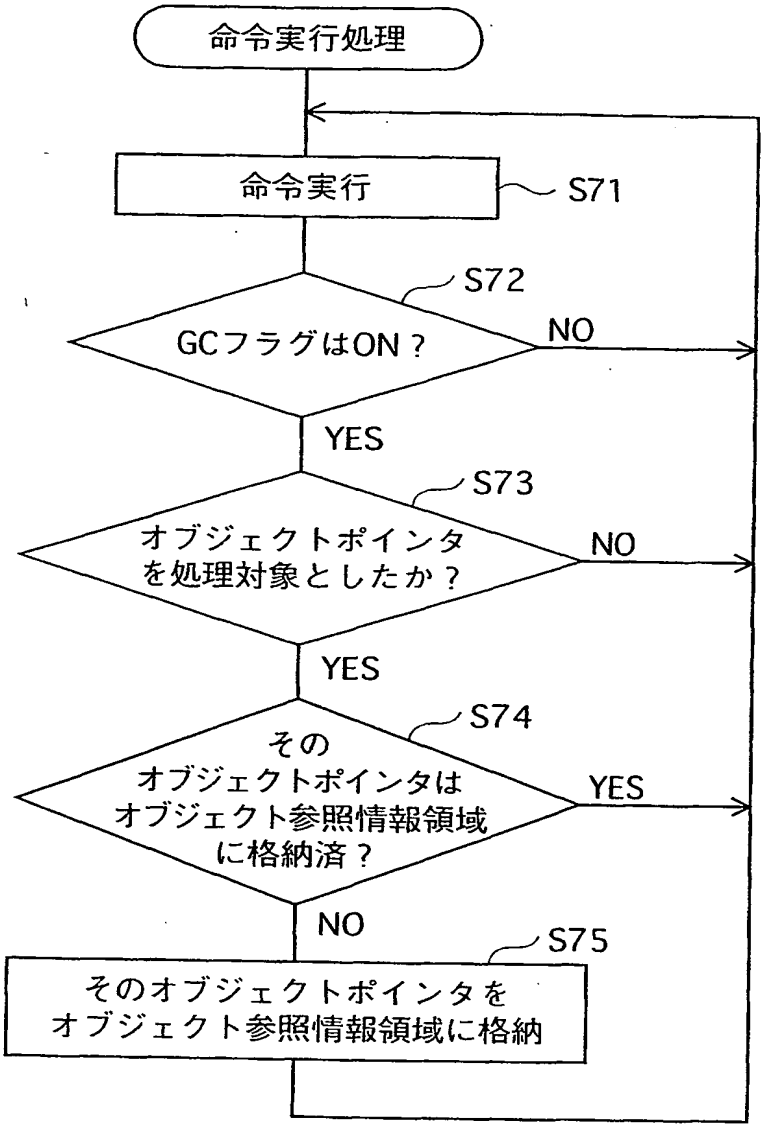
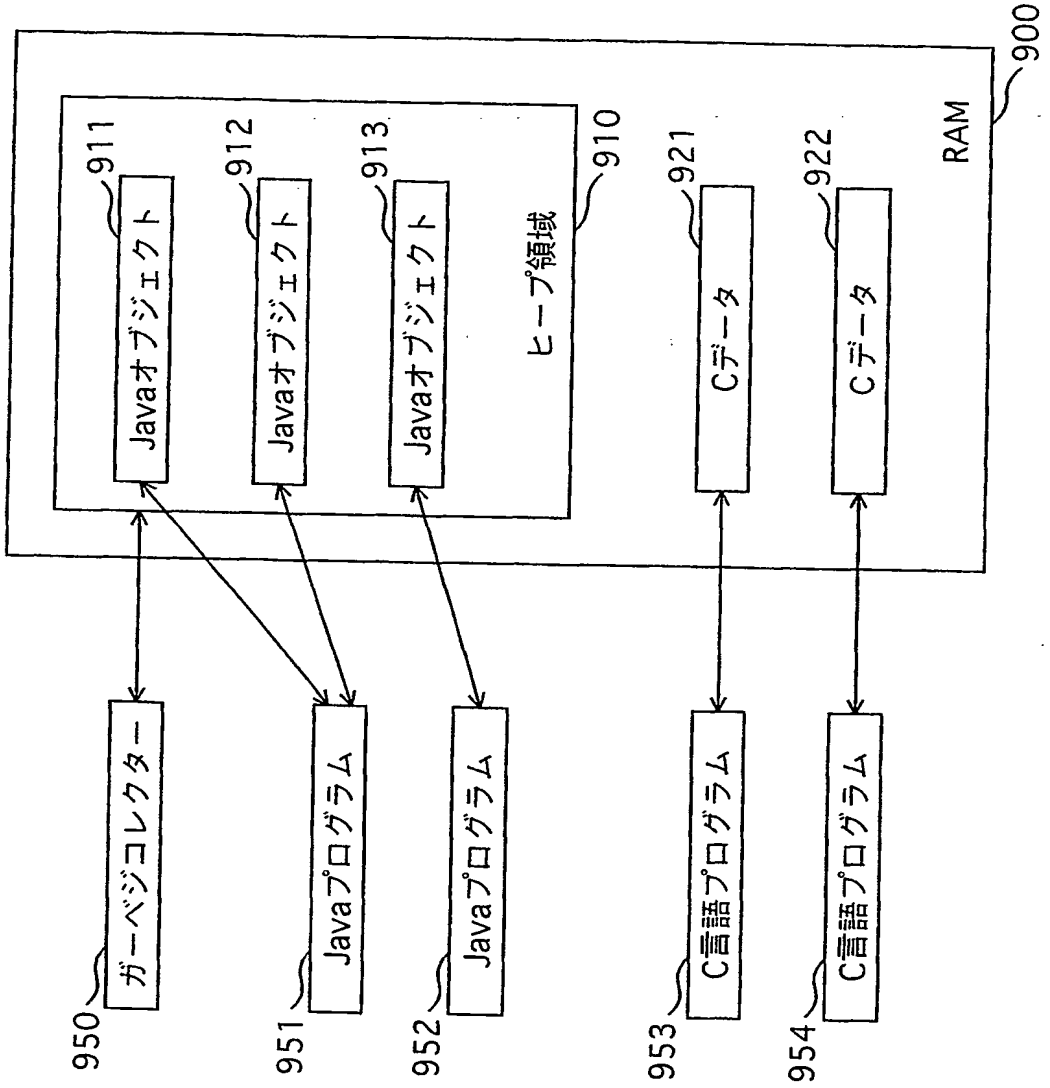

命令実行処理
命令実行
S71
S72
GCフラグはON？
NO
YES
S73
オブジェクトポインタ
を処理対象としたか？
NO
YES
S74
その
オブジェクトポインタは
オブジェクト参照情報領域
に格納済？
YES
NO
S75
そのオブジェクトポインタを
オブジェクト参照情報領域に格納

図19


900
RAM
910
ヒープ領域
911
Javaオブジェクト
912
Javaオブジェクト
913
Javaオブジェクト
921
Cデータ
922
Cデータ
950
ガーベジコレクター
951
Javaプログラム
952
Javaプログラム
953
C言語プログラム
954
C言語プログラム